PECAS - for Spatial Economic Modelling

Software User Guide

System Documentation Technical Memorandum 2
WORKING DRAFT

Template Materials Providing Basic Descriptive Components

JD Hunt, JE Abraham and D De Silva
HBA Specto Incorporated

Calgary, Alberta
February 2009

Table of Contents

List of Figures

List of Tables

"Computers make it easier to do a lot of things, but most of the things they make it easier to do don't need to be done."
Andy Rooney, 1919-

1. Introduction

1.1 Overview of PECAS

PECAS is a generalized approach for simulating spatial economic systems. It is designed to provide a simulation of the land use component of land use transport interactive modelling systems.

PECAS stands for Production, Exchange and Consumption Allocation System. Overall, it uses an aggregate, equilibrium structure with separate flows of exchanges (including goods, services, labour and space) going from production to consumption based on variable technical coefficients and market clearing with exchange prices. It provides an integrated representation of spatially distinct markets for the full range of exchanges, with the transport system and the development of space represented in more detail with specific treatments.

Flows of exchanges from production to exchange zones and from exchange zones to consumption are allocated using nested logit models according to exchange prices and transport generalized costs (expressed as transport utilities with negative signs). These flows are converted to transport demands that are loaded to networks in order to determine congested travel utilities. Exchange prices determined for space inform the calculation of changes in space thereby simulating developer actions. Developer actions are represented at either (a) the level of individual land parcels or grid cells using a microsimulation treatment or (b) the level of land use zones using an aggregate flow treatment. The system is run for each year being simulated, with the travel utilities and changes in space for one year influencing the flows of exchanges in the next year.

1.2 Purpose of This Document

This document is designed to describe the PECAS software and how to set-up and use the software. It outlines the software architecture and software operation, including descriptions of how the software works and instructions on how to run a PECAS model, specify inputs and obtain outputs. The contents of each input file and output file are listed.

This document does not describe the PECAS software and it does not provide specific guidance on how to either develop or run a PECAS model. Such descriptions and guidance are provided in other system documentation technical memoranda, as follows:

- PECAS Theoretical Formulation, System Documentation Technical Memorandum 1: for descriptions of the theoretical formulation of PECAS, including the concepts and the mathematical structure of the system and the processes used to provide representation of the relevant elements in the real-world system being simulated.
- PECAS Model Development and Calibration Procedures, System Documentation Technical Memorandum 3: for descriptions of the model design and development processes with PECAS, including aspects of data development and model calibration, drawing on what has evolved with the ongoing experience gained in previous model development efforts;

PECAS Software Programmer Reference, System Documentation Technical Memorandum 4.

1.3 Content of This Document

2. System Modules

PECAS includes two basic modules that are linked together with two other basic modules to provide a representation of the complete spatial economic system.

The set of four basic modules includes:
Space Development Module (SD Module): This is one of the two PECAS modules. It represents the actions of developers in the provision of different types of developed space where activities can locate, including the new development, demolition and re-development that occurs from one point in time to the next. This developed space is typically floorspace of various types and is called 'space' in the PECAS framework.
Activity Allocation Module (AA Module): This is the other of the two PECAS modules. It represents how activities locate within the space provided by developers and how these activities interact with each other at a given point in time.
Transport Model (TR Module): This is one of the 'non-PECAS' modules. It represents the transport system connecting locations, including at a minimum a transport network, the transport demands that load onto this network (as a result of the economic interactions represented in the AA Module) and the congested times and costs for interactions between locations arising with the loading of these demands. A standard, aggregate transportation planning model is likely to be sufficient.
Economic Demographic Aggregate Forecasting Model (ED Module): This is the other of the 'non-PECAS' modules: It is some form of model or approach used to develop aggregate economic forecasts for the study area being modeled. Typically, these forecasts include projected numbers of households or population by category and employment by type (as indications of expected economic activity) for specific points of time in the future. This model or approach may be able to adjust its forecasts in response to information from the AA and SD modules – as is represented in the descriptions included here – or it may provide a static set of forecasts. It may even be

the case that there is no model per se that is available, merely the forecast values for the study area. It is also possible to use an extended form of the PECAS AA Module to develop such aggregate forecasts – by making some specific assumptions about the relative contributions to the study-area economy from inside and outside the study area. For the descriptions included here, all of these possibilities are included in the single 'ED Module' designation that is used.

The four basic modules listed above are linked together with information flows as shown in Figure 1.

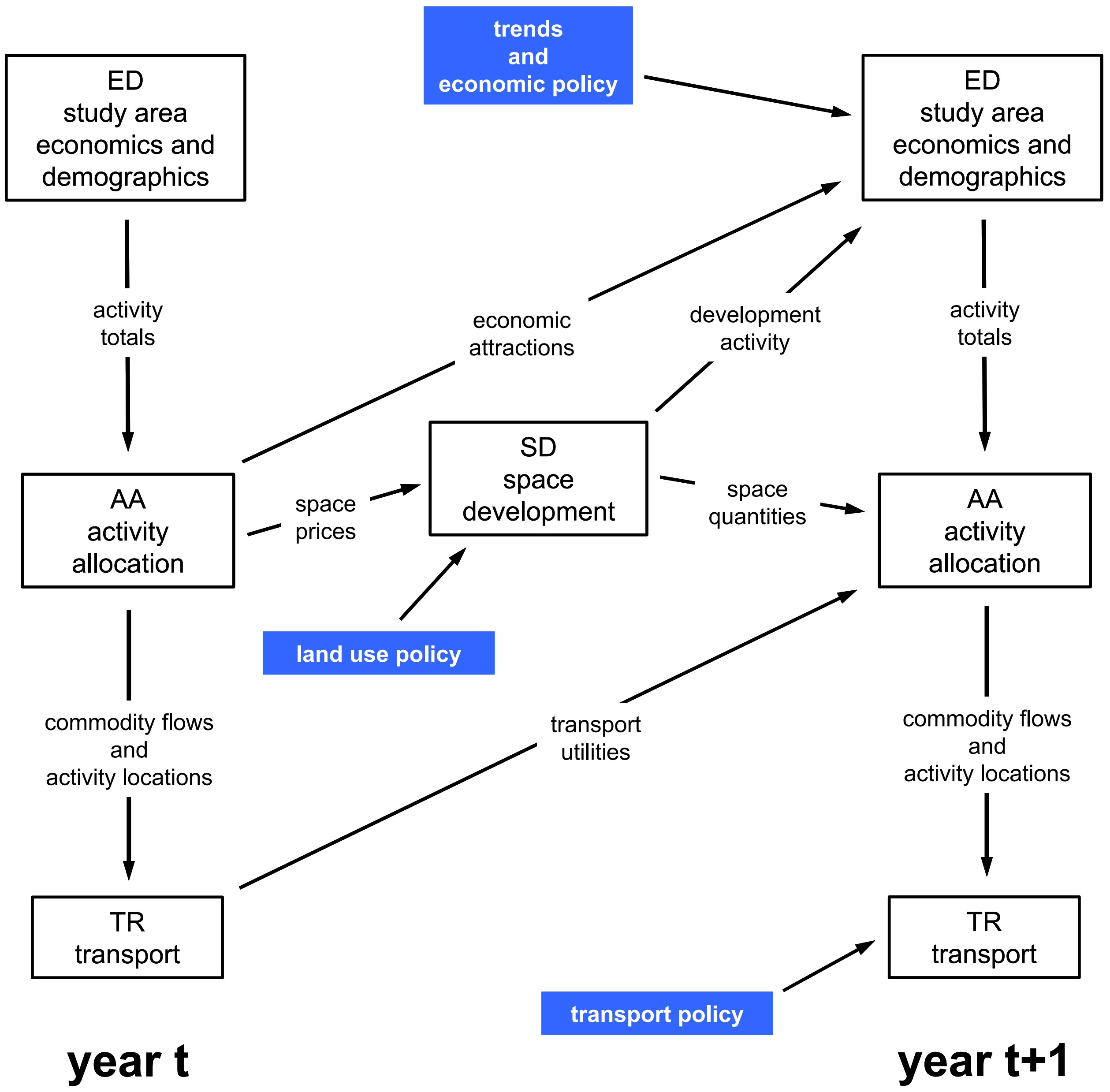


Figure 1: Modules and information flows simulating temporal dynamics

3. AA Module Architecture

3.1 AA Module Software Code

The AA Module is written in Java.

3.2 External Information Flows For AA Module

Figure 2 shows the information flows in and out of the AA Module in a given year t in the full set-up of the model system for running the simulation through time. It indicates the origins and destinations of these information flows and also upper-case letter codes that are used to identify specific files and variables included in these flows. Table 2 provides a summary of these flows and their components using the letter codes in Figure 2.

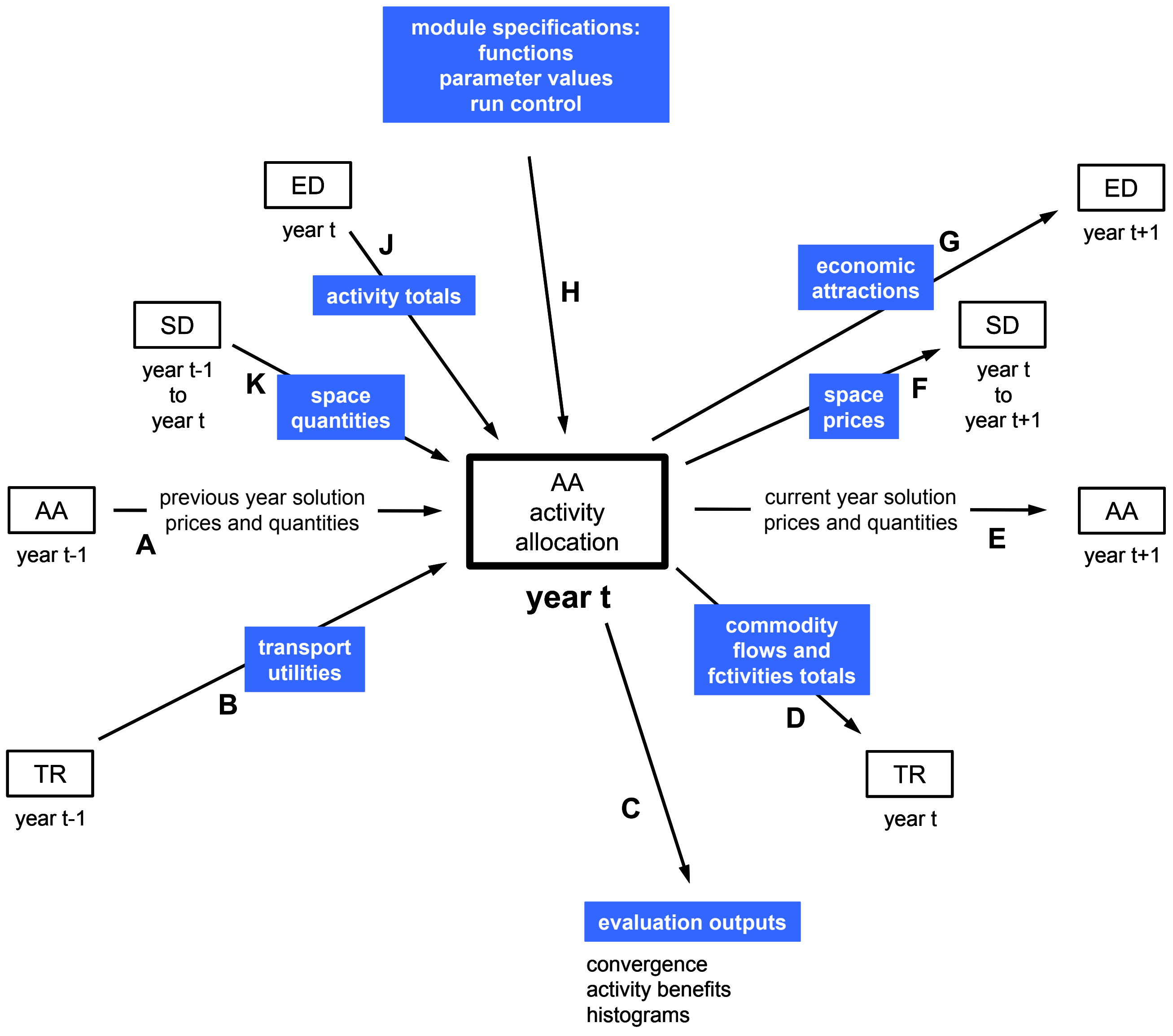


Figure 2: Information flows with specific input and output files for AA Module in the ‘full’ set-up where the AA Module is run in each year t in combination with the other system modules

Table 2: Component files and variables in AA Module information flows in Figure 2

Flow in Figure 2	File	Contents
A	ExchangeResults and ExchangeResultsI	Values for prices and exchange quantities for commodities; at the start of a model run the values in ExchangeResults or ExchangeResultsI are to be used as starting values
B	filename and directory location set in aa.properties using: skim.filename = filename and skim.data = directory location	Transport cost skims to be used
C	ActivityLocations	Activity quantities obtained by LUZ
C	ActivityResults	Activity quantities obtained by TAZ
C and G	ActivitySummary	Total model-wide quantities of activities obtained along with corresponding composite utilities.
C	CommodityZUtilities	Buying and selling quantities, composite utilities and related utility components obtained by commodity and LUZ
C	ConsumptionErrorTermSizes	Values obtained for components of consumption utilities by activity
C	ExchangeResultsTotals	Summary of model-wide values for prices and exchange quantities obtained by commodity

C	Histograms	Histograms presenting run results using format(s) as specified
C	LatestActivityConstraints	Activity quantities obtained by LUZ at an intermediate point before the model run is finished
C	PctIntrazonalxCommodityxBZone	Summary of flow allocation components obtained by commodity and exchange location
C	ProductionErrorTermSizes	Values obtained for components of production utilities by activity
C	ZonalMakeUse	Production and consumption quantities obtained by commodity and LUZ
D	Buying_*CommodityName*	Origin-Destination flows to selling locations obtained by commodity
D	Selling_*CommodityName*	Origin-Destination flows from buying locations obtained by commodity
E and F	ExchangeResults	Values for prices and exchange quantities for commodities; at the end of a model run the values in ExchangeResults are outputs obtained from a completed model run
H	ActivitiesI	Definitions of activity names and values to be used for activity parameters

H	ActivityConstraintsI	Activity quantities to be obtained by LUZ; for use in calibration to target values and also for constraining to exogenously determined values
H	ActivitySizeTermsI	Space types and weights to be used in establishing size terms for activity allocation
H	ActivitiesZonalValuesI	Values for constants, size terms and initial quantities to be used for activity allocation
H	CommoditiesI	Definitions of commodity names and values to be used for commodity parameters
H	ExchangeImportExportI	Values to be used for commodity imports and exports function parameters
H	FloorspaceBuyingSizeTermsI	Space types and weights to be used in establishing size terms for commodity allocations
H	FloorspaceSupplyI	Values to be used for space supply functions
H	FloorspaceZonesI	Listing of TAZ in each LUZ
H	HistogramsI	Specification of histograms of commodity flow distributions to be output
H	PECASZonesI	Definitions of LUZ

H	TechnologyOptionsI	Definitions of technology option points to be used in technology allocations
J	ActivityTotals and ActivityTotalsI	Total model-wide quantities of activities to be allocated among LUZ
K	Floorspace and FloorspaceI	Space quantities by type and LUZ to be used for activity allocation

3.3 AA Module Inputs

Input Tables System

The inputs to the AA Module are quantities for variables specified in either individual .csv files or individual tables in a jdbc compatible data source. There are special provisions allowing Excel to be used as a jdbc data source, so that its worksheets can be used as input tables. The column headers and format for a given table are the same in all cases. Except where indicated otherwise, the term ‘input table’ is used here to refer to all cases.

Each of the input tables has a specific name, which the AA Module uses to identify the type of table it is and thereby establish the nature of its contents. For input tables contained in individual .csv files, the table name is the name of the file before the .csv extension. For input tables contained in individual sheets in an Excel file, the table name is the name of the individual worksheet (typically shown in the tab at the bottom-left corner of the spreadsheet window).

The linkages to the input tables for a run of the AA Module are set in the aa.properties file – including the directory containing the .csv files, database files or Excel spreadsheet files and the filenames for database or Excel spreadsheets as required.

Input tables prepared by the user (rather than by another module in the modelling system) have names that end with an upper-case 'I'. Certain input tables are sometimes prepared by another module and sometimes prepared by the user, including ActivityTotals (or ActivityTotalsI), Floorspace (or FloorspaceI), and ExchangeResults (or ExchangeResultsI). If both are present in the specified directory, then the input table with an upper-case 'I' is used. Having multiple sources for certain input files facilitates the use of two kinds of 'set-up' for the AA module, a 'separate' set-up used at a certain stage in model development and a 'full' set-up used at another stage in model development and in model application.

When input tables prepared by the user are first modified by a pre-processor before being read into the AA Module, then the processed versions of the input tables have names that end with an upper-case 'W' rather than an upper-case 'I', but with the rest of the name remaining the same. That is, the input table 'CommoditiesI.csv' is a file prepared by the user, whereas the input table 'CommoditiesW'csv' is the file resulting from the pre-processing of 'CommoditiesI.csv' before it is read into the AA Module. Note that efforts are being made to phase-out the use of files with names ending with an upper-case 'W'.

The AA Module first reads the input tables contained in individual sheets in any specified database or Excel spreadsheet file. It then reads any input .csv files in the indicated directories, using the values in these .csv files in place of the corresponding values in the database or Excel file. When this happens (where values in .csv files are used to replace other values) a warning is written to the log file. There are two indicated directories for these input .csv files, one set to aa.base.data and the other set to aa.current.data.

The AA Module looks in the directory set to aa.base.data for the following tables:
CommoditiesI;
ExchangeImportExportI;
ActivitySizeTermsI;
FloorspaceBuyingSizeTermsI;
HistorgramsI;
TechnologyOptionsI;
ActivitiesI;
ActivityZonalValuesI; and
FloorspaceI
(all with .csv extensions).

The AA Module looks in the directory set to aa.current.data for the following tables:
ActivitiesW;
ActvityZonalValuesW;
ExchangeResultsI; and
ActvityConstraintsI
(again, all with .csv extensions).

These settings for the indicated directories are specified in the aa.properties file.

Content of Input Tables

The separate input tables for the AA Module are:

- ActivitiesI; each row concerns a specific activity, indicating the name and parameter values for the activity

  - Activity; text characters of name of activity; this is used as the definitive name for the activity throughout the AA Module;
  - LocationDispersionParameter; real value of dispersion parameter for location allocation utility function for activity; this is $\lambda_{l,a}$ in equation 02 in the Theoretical Formulation document;
  - SizeTermCoefficient; real value of coefficient on size term in location allocation utility function for activity; this is $\alpha_{size,a}$ in equation 02 in the Theoretical Formulation document;
  - ProductionSubstitutionNesting; dispersion parameter for technology option utility function used to allocate activity among technology options; this is $\lambda_{p,a}$ in equations 03 and 04 in the Theoretical Formulation document;
  - InertiaTermCoefficient; this is fixed at 0 in the software code; it may be included in a future version and specified here;
  - InertiaTermConstant; this is fixed at 0 in the software code; it may be included in a future version and specified here;

- ActivityConstraintsI; each row concerns a specific activity in a specific land use zone (LUZ), indicating the quantity of the activity to be allocated to the LUZ by the AA Module when using the activity constraints routine, which is described in section 4.7 of this document;
  - BZone; integer value of land use zone (LUZ);
  - Activity; text characters of name of activity; this must match exactly the activity name used in ActivitiesI;
  - Quantity; real value of quantity of activity to be allocated to the LUZ;

- ActivitySizeTermsI; this is used to specify the space types and weights for the AA Module to use with the ‘exogenous sizes’ approach to establishing the values for the size terms in the utility functions for allocating activities to zones, which is described in section 4.5 of this document; each row concerns a specific combination of activity and space type, indicating the weight to use for the space type in the calculation of the size terms for the activity, as described below in this document;

  - Activity; text characters of name of activity; this must match exactly the activity name used in ActivitiesI;
  - Weight; real value of weight to use with space type in the calculation of weighted sum of space total for each zone for the activity; this is the value of $SpSWgt_{c,a}$ for the activity, indexed a, and the space type, indexed c, indicated in the description of the 'endogenous sizes' approach included in Section 4.5 in this document;
  - Floorspace; text character name of space type; this must match exactly the commodity name used for this space type in its representation as a commodity in the AA Module; this must match exactly the commodity name used in CommoditiesI;

- ActivityTotalsI; each row concerns a specific activity, indicating the name and total model-wide quantity for the activity
  - TotalAmount; total model-wide quantity of activity to be allocated among zones; this is TW_a in equations 01 and 03 in the Theoretical Formulation document;

- ActivitiesZonalValuesI; each row concerns a specific activity in a specific zone, indicating the values to use for the initial quantity, the size term and the zone specific constant for the location utility for the activity in the zone
  - Activity; text characters of name of activity; this must match exactly the activity name used in ActivitiesI;
  - ZoneNumber; integer value of land use zone (LUZ);
  - InitialQuantity; real value of the proportion of the model-wide quantity of activity in zone in the previous time period; this is $PrevW_{a,z}$ in equation 02 in the Theoretical Formulation document.
  - SizeTerm; real value of size term in location utility function; this is $Size_{a,z}$

in equation 02 in the Theoretical Formulation document; values for size terms are specified here with the 'exogenous size' approach to establishing the values for the size terms in the utility functions for allocating activities to zones, which is described in Section 3.8 in this document; these values are ignored if the 'endogenous size' approach is used as described in Section 3.8;

  - ZoneConstant; real value of zone specific constant in location utility function; this is $Constant_{a,z}$ in equation 02 in the Theoretical Formulation document;

- CommoditiesI; each row concerns a specific commodity (category of goods, services, labour or space), indicating the name and parameter values for the commodity
  - Commodity; text characters of name of commodity; this is used as the definitive name for the commodity throughout the AA and SD Modules;
  - BuyingDispersionParameter; real value of dispersion parameter for utility function for allocation of buying of commodity; this is λb_c in equations 16, 17 and 18 in the Theoretical Formulation document;
  - SellingDispersionParameter; real value of dispersion parameter for utility function for allocation of selling of commodity; this is λs_c in equations 12, 13 and 14 in the Theoretical Formulation document;
  - BuyingSizeCoefficient; real value of utility function coefficient for the sensitivity to size when buying commodity; this is $\delta b_{xsize,c}$ in equation 17 in the Theoretical Formulation document;
  - BuyingPriceCoefficient; real value of utility function coefficient for the sensitivity to price when buying commodity; this is $\delta b_{price,c}$ in equation 17 in the Theoretical Formulation document;
  - BuyingTransportCoefficient; real value of utility function coefficient for the sensitivity to transport when buying commodity; this is $\delta b_{tran,c}$ in equation

17 in the Theoretical Formulation document;

- SellingSizeCoefficient; real value of utility function coefficient for the sensitivity to size when selling commodity; this is $\delta s_{xsize,c}$ in equation 13 in the Theoretical Formulation document;
- SellingPriceCoefficient; real value of utility function coefficient for the sensitivity to price when selling commodity; this is $\delta s_{price,c}$ in equation 13 in the Theoretical Formulation document;
- SellingTransportCoefficient; real value of utility function coefficient for the sensitivity to transport utility when selling commodity; this is $\delta s_{tran,c}$ in equation 13 in the Theoretical Formulation document;
- InterchangeCoefficient1; real value for interchange attribute 1, used to calculate the utility for transporting one unit of commodity from land use zone (LUZ) to land use zone (LUZ), in units of transport utility per commodity unit; this is $\kappa 1_c$ in equation 20 in the Theoretical Formulation document;
- InterchangeCoefficient2; real value for interchange attribute 2, used to calculate the utility for transporting one unit of commodity from land use zone (LUZ) to land use zone (LUZ), in units of transport utility per commodity unit; this is $\kappa 2_c$ in equation 20 in the Theoretical Formulation document;
- InterchangeCoefficient3; real value for interchange attribute 3, used to calculate the utility for transporting one unit of commodity from land use zone (LUZ) to land use zone (LUZ), in units of transport utility per commodity unit; this is $\kappa 3_c$ in equation 20 in the Theoretical Formulation document;
- InterchangeName1; text characters of name for interchange attribute 1, used to calculate the utility for transporting one unit of commodity from land use zone (LUZ) to land use zone (LUZ); this must match exactly the name used for interchange attribute 1 in its column in the file containing

the transport conditions for movements from land use zone (LUZ) to land use zone (LUZ) (the skims) output by the TR Module (the transport model), the name and directory location of this file specified using the skim.data and skim.filename program variables in the aa.properties file as described in section 6 of this document;

- InterchangeName2; text characters of name for interchange attribute 2, used to calculate the utility for transporting one unit of commodity from land use zone (LUZ) to land use zone (LUZ); this must match exactly the name used for interchange attribute 2 in its column in the file containing the transport conditions for movements from land use zone (LUZ) to land use zone (LUZ) (the skims) output by the TR Module (the transport model), the name and directory location of this file specified using the skim.data and skim.filename program variables in the aa.properties file as described in section 6 of this document;
- InterchangeName3; text characters of name for interchange attribute 3, used to calculate the utility for transporting one unit of commodity from land use zone (LUZ) to land use zone (LUZ); this must match exactly the name used for interchange attribute 3 in its column in the file containing the transport conditions for movements from land use zone (LUZ) to land use zone (LUZ) (the skims) output by the TR Module (the transport model), the name and directory location of this file specified using the skim.data and skim.filename program variables in the aa.properties file as described in section 6 of this document;
- ExchangeType; single text character indicating the exchange regime for commodity, with possible values:

‘c’	=	commodity is exchanged only in consumption zones (where the seller does all transporting)
‘p’	=	commodity is exchanged only in production zones (where the buyer does all transporting)
‘a’	=	commodity exchanged in any zone (where both buyer and seller may do some of the transporting)

'n'	=	commodity is non-transportable (where the commodity is consumed in the same zone where it is produced); this is the value for each space type included as a commodity;
's'	=	commodity is exchanged only in specified zones (both buyer and seller may do some of the transporting, but exchanges occur in only certain zones); for this regime the only land use zones where exchanges can occur are specified by a value of 'true' for SpecifiedExchange in the ExchangeImportExportI file.

This is $ExCh_c$ described in section 5.3 of the Theoretical Formulation document;

- GOFWeighting; weight applied to residual-squared values for commodity when summing residual-squared values across commodities to assess extent of convergence to market clearing solution by AA Module; this is $VergeWt_c$ in equation 35 in the Theoretical Formulation document;
- FloorspaceCommodity; text characters indicating whether commodity is a space type; 'true' indicate the commodity is a space type, 'false' indicates the commodity is not a space type;
- ExpectedPrice; real value of price to use for commodity in exchange zone as a starting price if there is no ExchangeResults or ExchangeResultsI file to provide starting prices. Also used in the endogenous calculation of buying and selling size terms for non-floorspace commodities (section 3.8).
- Units; text characters of units used for expressing quantities of commodity;

- ExchangeImportExportI; each row concerns the import and export functions and the exchange potential for a specific commodity in a specific exchange zone, indicating the parameter values for the functions, and also whether to write interim values for the exchange to the log file in order to monitor the AA Module solution algorithm; note that a value of '-1' for the zone number indicates that the values are for the commodity in all the exchange zones not considered in a separate row for the commodity; also note that space commodities cannot be imported or exported and thus cannot be considered in this spreadsheet, they are considered in the FloorspaceSupplyI spreadsheet described immediately below;

- ZoneNumber; integer value of land use zone (LUZ) that is the exchange location zone; a value of -1 indicates all land use zones not considered explicitly in other rows using single zone numbers in single rows for the commodity
- Commodity; text characters of name of commodity; this must match exactly the commodity name used in CommoditiesI;
- BuyingSize; real value representing the relative size of exchange zone for buying commodity, indicating the *a priori* expected share of commodity bought in the exchange zone; this is $XbSize_{c,k}$ in equation 17 in the Theoretical Formulation document;
- SellingSize; real value representing the relative size of exchange location for selling commodity, indicating the *a priori* expected share of commodity sold in exchange location; this is $XsSize_{c,k}$ in equation 13 in the Theoretical Formulation document
- SpecifiedExchange; text characters 'true' indicate the exchange zone is a specified exchange zone for commodity, otherwise the exchange zone is not a specified exchange zone; the commodity must be designated to be exchanged only in specified zones using a value of 's' for the ExchangeType for the commodity in the CommoditiesI file; this is described in section 5.3 of the Theoretical Formulation document;
- ImportFunctionMidpoint; real value of import function quantity of commodity imported to exchange location when the unit exchange price for commodity in exchange zone is at its import reference level $PriceIRef_c$; this is $QiRef_c$ in equation 21 in the Theoretical Formulation document;
- ImportFunctionMidpointPrice; real value of import function reference price per unit for import of commodity; this is $PriceIRef_c$ in equations 21 and 23

in the Theoretical Formulation document;

- ExportFunctionMidpoint; real value of export function quantity of commodity exported from exchange location when the unit exchange price for commodity in exchange zone is at its export reference level $PriceERef_c$; this is $QeRef_c$ in equation 22 in the Theoretical Formulation document;
- ExportFunctionMidpointPrice; real value of export reference price per unit for export of commodity; this is $PriceERef_c$ in equations 22 and 24 in the Theoretical Formulation document;
- ImportFunctionDelta; real value of import function coefficient for the rate of increase in imports of commodity for exponent term; this is Δi_c in equation 21 in the Theoretical Formulation document;
- ImportFunctionSlope; real value of import function coefficient for the rate of increase in imports of commodity for linear term; this is μi_c in equation 21 in the Theoretical Formulation document;
- ImportFunctionEta; real value of import function coefficient for sensitivity to difference in exchange price for commodity concerning increase in imports of commodity for exponent term; this is ηi_c in equation 23 in the Theoretical Formulation document;
- ExportFunctionDelta; real value of export function coefficient for the rate of increase in exports of commodity for exponent term; in software: this is Δe_c in equation 22 in the Theoretical Formulation document;
- ExportFunctionSlope; real value of export function coefficient for the rate of increase in exports of commodity for linear term; this is μe_c in equation 22 in the Theoretical Formulation document;
- ExportFunctionEta; real value of export function coefficient for sensitivity to difference in exchange price for commodity concerning increase in exports of commodity for exponent term; this is ηe_c in equation 24 in the Theoretical Formulation document;
- MonitorExchange; text characters indicating whether interim values for the

commodity in the exchange zone are to be written to the log file in order to monitor the AA Module solution algorithm; 'true' indicates the interim values are to be written, 'false' indicates the interim values are not to be written;

- ExchangeResults; each row concerns a specific commodity and exchange location land use zone (LUZ), indicating the starting values to use in a run of the AA Module; the ExchangeResults.csv file from a previous run of the AA Module can be used directly if desired; in general, it is only the values for the unit prices for the commodities in the exchange zones established in a previous run, the $\text{Price}_{c,k}$ values, that are useful in the start of another run, but the full format of the ExchangeReults.csv file is accepted in ExchangeResultsI;
  - Commodity; text characters of name of commodity; this must match exactly the commodity name used in CommoditiesI;
  - ZoneNumber; integer value of exchange location land use zone;
  - Demand; real value of the demand for the commodity in the exchange location, in units of the commodity; this is $\text{TDem}_{c,k}$ in equation 30 in the Theoretical Formulation document;
  - InternalBought; real value of internal bought quantity of commodity exchange location, this is the total quantity of the commodity being bought in the exchange zone by all activities internal to the model area, in units of the commodity; this is $\text{TBInt}_{c,k}$ in equation 28 in the Theoretical Formulation document;
  - Exports; real value of quantity of commodity c exported from exchange location k; this is $\text{Qe}_{c,k}$ in equations 22 and 30 in the Theoretical Formulation document;
  - Supply; real value of the demand for the commodity in the exchange location, in units of the commodity; this is $\text{TSup}_{c,k}$ in equation 31 in the Theoretical Formulation document;

  - InternalSold; real value of internal sold quantity of commodity in exchange location, this is the total quantity of the commodity being sold in the exchange zone by all activities internal to the model area, in units of the commodity; this is $TSInt_{c,k}$ in equation 29 in the Theoretical Formulation document;
  - Imports; real value of quantity of commodity c imported to exchange location k; this is $Qi_{c,k}$ in equations 21 and 13 in the Theoretical Formulation document;
  - Surplus; real value of the residual surplus quantity of supply minus demand for the commodity in the exchange location, in units of the commodity; this is $Residual_{c,k}$ in equation 33 in the Theoretical Formulation document;
  - Price; real value of unit exchange price for commodity in exchange location, in units of money per unit of the commodity; this is $Price_{c,k}$ in the Theoretical Formulation document.

- FloorspaceI; each row concerns a specific space type (corresponding to a commodity category in the AA Module) in a particular transport model zone (TAZ), indicating the value for the available quantity to use in a run of the AA Module; the Floorspace.csv file from a previous run of the SD Module is used in the full setup; the values for the TAZ are aggregated to establish the values for the corresponding LUZ by the AA Module as described in section 4.6 in this document;
  - FloorspaceZone; integer value of transport analysis zone (TAZ);
  - Commodity; text characters of name of the commodity category used to represent the space type in the AA Module; this must match exactly the commodity name used in CommoditiesI;
  - Quantity; real value of physical quantity of space type in transport analysis zone (TAZ); this is determined in the SD Module and input to the AA

Module, included in the 'space quantities' information flow shown in Figure 1; this is $TSpcQty_{c,i}$ and the values across the TAZ in an LUZ are summed to establish $SpcQty_{i,z}$ in equation 25 in the Theoretical Formulation document.

- FloorspaceBuyingSizeTermsI; each row concerns a specific space type (corresponding to a commodity category in the AA Module), indicating the space group to which the space type belongs for the purpose establishing size terms for the allocation of buying quantities of the space type; In the Theoretical Formulation Document, $XbSize_{c,k}$ in equation 17 is set to the quantity of space in of type c in each zone k (from FloorspaceI) if the space type appears in this file and the FloorspaceGroup is set to "none". If the space type appears in this file, and "FloorspaceGroup" is set to something other than "none", the $XbSize_{c,k}$ in equation 17 is set to $Scale_c*Flr_{c,k}/\sum_{c1 \in group}(Scale_{c1}*Flr_{c1,k})$. Space groups should not be set to "none" if Activity Size Terms are specified in ActivitySizeTermsI, since that would lead to a double accounting of the actual zone size (once here at the exchange choice level, and once at the activity zone choice level.)
    - FloorspaceType; text characters of name of the commodity category used to represent the space type in the AA Module; this must match exactly the commodity name used in CommoditiesI;
    - FloorspaceGroup; text characters of name of the space group;
    - Scale; relative adjustment factor for floorspace type.

- FloorspaceSupplyI; each row concerns the supply function for a specific space type (corresponding to a commodity category in the AA Module) in a specific land use zone, indicating the parameter values for the function and whether to provide a monitor of the supply quantity in the log file; note that a value of '-1' for the zone number indicates that the values are for the space type for all the land use zones not considered in a separate row for the space type;

- ZoneNumber; integer value of land use zone (LUZ) that is the exchange location zone; a value of -1 indicates all land use zones not considered explicitly in other rows using single zone numbers in single rows for the commodity
- Commodity; text characters of name of the commodity category used to represent the space type in the AA Module; this must match exactly the commodity name used in Commodities**I**;
- SupplyFunctionMidpointFactor; real value of proportion of space for space type available as a commodity when the unit exchange price for the space is at its reference level $PriceBas_h$; this is $SpcBas_h$ in equation 26 in the Theoretical Formulation document;
- SupplyFunctionMidpointPrice; real value of reference base price per unit for space type for the proportion of space available as a commodity; this is $SpcBas_h$ in equations 26 and 27 in the Theoretical Formulation document;
- SupplyFunctionDeltaFactor; real value of space supply function coefficient for the rate of increase in the proportion of space available as a commodity for exponent term in space supply function; this is Δ_h in equation 26 in the Theoretical Formulation document;
- SupplyFunctionSlopeFactor; real value of space supply function coefficient for the rate of increase in the proportion of space available as a commodity for linear term in space supply function; this is μ_h in equation 26 in the Theoretical Formulation document;
- SupplyFunctionEta; real value of space supply function coefficient for sensitivity to difference in exchange price for space type, concerning increase in the proportion of space available as a commodity for linear term in space supply function; this is η_h in equation 27 in the Theoretical Formulation document;
- NoQuantityPrice; real value of unit price to use for space type in zone when there is no supply of the space and the price update process in the

solution algorithm in the AA Module cannot establish an appropriate price value as a result;

- NoQuantitySlope; real value of slope to use in the price update function for space type in zone when there is no supply of the space and the price update process in the solution algorithm in the AA Module cannot establish an appropriate price value as a result;
- MonitorExchange; text characters indicating whether interim values for the space type in the zone are to be written to the log file in order to monitor the AA Module solution algorithm; 'true' indicates the interim values are to be written, 'false' indicates the interim values are <u>not</u> to be written.

- FloorspaceZonesI; each row concerns a specific transport analysis zone (TAZ), indicating the TAZ and the land use zone (LUZ) containing the TAZ;
  - TAZone; integer value of transport analysis zone (TAZ);
  - AAZone; integer value of land use zone (LUZ) containing the TAZ.

- HistogramsI; each row specifies a histogram to be output showing the distribution of transport lengths for the zone-to-zone flows established for a commodity in the AA Module;
  - Commodity; text characters of name of commodity to show in the histogram; this must match exactly the commodity name used in CommoditiesI;
  - Skim; text characters of name of interchange attribute to use to define ranges of transport lengths to use in histogram; this must match exactly the name used for the interchange attribute in its column in the file containing the transport conditions for movements from land use zone (LUZ) to land use zone (LUZ) (the skims) output by the TR Module (the transport model), the name and directory location of this file specified using the skim.data and skim.filename program variables in the aa.properties file as described in section 6 of this document;

- C0; real value of interchange attribute defining boundary between first and second ranges of transport lengths to use in histogram;
- C1; real value of interchange attribute defining boundary between second and third ranges of transport lengths to use in histogram;

  and so on
- C*n*; real value of interchange attribute defining boundary between (*n*+1)th and (*n*+2)th ranges of transport lengths to use in histogram;

  until *n* is 14 and defining the boundary between the 15th and 16th ranges of transport lengths to use in histogram.

- PECASZonesI; each row concerns a specific land use zone (LUZ), indicating the zone number and the zone name for the land use zone
  - ZoneNumber; integer value of land use zone (LUZ);
  - ZoneName; text characters of name of land use zone (LUZ).
  - External; ‘true’ or ‘false’ indicating whether the zone is an external zone. (Flows are not allowed between external zones.)

- TechnologyOptionsI; each row concerns a specific technology point option for a particular activity, indicating the vector of production rates for the set of commodities made and the vector of consumption rates for the set of commodities consumed for that technology point
  - Activity; text characters of name of activity; this must match exactly the activity name used in ActivitiesI;
  - OptionName; text characters for name of technology point
  - OptionSize; real value for coefficient on option size term in technology allocation utility function for activity; this is $OptionSize_{p,a}$ in equation 04 in the Theoretical Formulation document;
  - Vector of production rates corresponding to column headers indicating commodities produced in each case; numerical value for rate of production of commodity by activity; units are produced commodity units

per activity unit; this is $ProdRate_{p,a,c}$ in equation 05 in the Theoretical Formulation document;

- Vector of consumption rates corresponding to column headers indicating commodities consumed in each case with a ':1' appended to the header in each case; negative numerical value for rate of consumption of commodity by activity; units are consumed commodity units per activity unit; this is $ConsRate_{p,a,c}$ in equation 06 in the Theoretical Formulation document.

3.4 AA Module Outputs

The structure and contents of the separate output files for the AA Module are indicated below.

- ActivityLocations; each row concerns a specific activity and land use zone (LUZ), indicating the components of the location utility and resulting quantity of activity allocated to the zone in a run of the AA Module;
    - Activity; text characters of name of activity as specified in ActivitiesI;
    - ZoneNumber; integer value for the land use zone (LUZ)
    - Quantity; real value of the quantity of activity in the LUZ, which is $W_{a,z}$ in equations 01 and 03 in the Theoretical Formulation document
    - TechnologyLogsum; real value for the composite utility associated with the range of technology options for activity in zone; this is in general $CUTech_{a,z}$ in equation 02 in the Theoretical Formulation document
    - SizeUtility; real value for the product of the natural log of the size term, the corresponding size term coefficient and the inverse of the dispersion parameter in the location utility function for the activity in the zone; this is in general the product $\alpha_{size,a} \cdot 1/\lambda_{l,a} \cdot \ln [Size_{a,z}]$ included in equation 02 in the Theoretical Formulation document;
    - LocationSpecificUtility; real value for the utility function zonal alternative

specific constant for activity for zone; this is in general Constant$_{a,z}$ in equation 02 in the Theoretical Formulation document; if the 'activity constraints' routine is activated then the values for the Constant$_{a,z}$ are those achieved by the routine as described below; if the 'activity constraints' routine is not activated then the values for the Constant$_{a,z}$ are those specified as inputs in ActivityZonalValues**I**;

  - LocationUtility; real value of location utility for a unit of activity in zone; this is in general LU$_{a,z}$ in equation 02 in the Theoretical Formulation document ;

- ActivitySummary; each row concerns a specific activity and its model-wide total size and location composite utility in a run of the AA Module;
  - Activity; text characters of name of activity as specified in Activities**I**;
  - CompositeUtility; numerical value for model-wide location composite utility for activity, in utils per unit of activity; this is CLU$_a$ in equation 31 in the Theoretical Formulation document;
  - Size; numerical value for model-wide size of the activity in the units specified for the activity; this is TW$_{a,z}$ in equation 01 in the Theoretical Formulation document.

- Buying_*CommodityName*; origin-destination matrix of flows of commodity with text name '*CommodityName*' going from exchange zone (LUZ) where it is bought to location zone (LUZ) where it is consumed; row numbers are origin (exchange) zones and column numbers are destination (consumption) zones; each cell in the matrix indicates the quantity of flow of the commodity in the specified units for the commodity determined in a run of the AA Module, B$_{c,z,k}$ in equation 16 in the Theoretical Forumulation Document;

  The file may be written in zipMatrix format, which will be indicated by its filename extension; to view a file in zipMatrix format the MatrixViewer program must be run; this is done by clicking on the viewmatrix.cmd icon in the base year directory (and perhaps elsewhere) which will open a cmd window and also start the MatrixViewer program (current version 0.4); the file tab in the MatrixViewer window can then be used to open and view the file; the cell values can then be imported to an Excel spreadsheet if desired using copy (Control-C) and paste (Control-P) commands

- CommodityZUtilities; each row concerns a specific commodity and its buying or selling composite utility for a specific location LUZ, if for buying then indicating the quantity consumed in the location zone and the buying composite utility and its components or if for selling then indicating the quantity produced in the location zone and the selling composite utility and its components in a run of the AA Module;
  - Commodity; text characters of name of commodity as specified in CommoditiesI;
  - Zone; integer number of LUZ; this is a in the Theoretical Formulation document;
  - BuyingOrSelling; code letter indicating if values in row are for buying (code letter ‘B’) or for selling (code letter ‘S’)
  - Quantity; if buying, then this is the amount of commodity bought anywhere and brought to zone for consumption; this is $TC_{c,z}$ in equation 11 in the Theoretical Formulation document; if selling, then this is the amount of commodity produced in zone and then sold anywhere; this is $TP_{c,z}$ in equation 09 in the Theoretical Formulation document;
  - zUtility; utility in utils per unit of commodity; CUSellc,z in equation 14 in the Theoretcial Formulation Document or CUBuyc,z in equation 18.
  - PriceComponent; $1/TC_{c,z} \sum_{k \in K} B_{c,z,k} * \delta b_{price,c} * Price_{c,k}$, the utility effect of the average price paid when located in zone z and buying commodity c (if buying) or $1/TP_{c,z} \sum_{k \in K} S_{c,z,k} * \delta s_{price,c} * Price_{c,k}$, the utility effect of the average price earned when located in zone z and selling commodity c (if

selling).(only appears if aa.writeUtilityComponents=true in the properties file.)

- SizeComponent; $1/TC_{c,z} \sum_{k\in K} B_{c,z,k} * \delta b_{size,c} * 1/ \lambda s_c * \ln[XsSize_{c,k}]$, the utility effect of the size of the zones available and chosen when located in zone z and buying commodity c (if buying) or $1/TP_{c,z} \sum_{k\in K} S_{c,z,k} * \delta s_{size,c} * 1/ \lambda b_c * \ln[XbSize_{c,k}]$, the utility effect of the size of the zones available and chosen when located in zone z and selling commodity c (if buying); (only appears if aa.writeUtilityComponents=true in the properties file.)
- TransportComponent; $1/TC_{c,z} \sum_{k\in K} B_{c,z,k} * \delta b_{trans,c} * TRAN_{c,k,z}$, the utility effect of transport when located in zone z and buying commodity c (if buying) or $1/TP_{c,z} \sum_{k\in K} S_{c,z,k} * \delta s_{trans,c} * TRAN_{c,z,k}$, the utility effect of transport when located in zone z and selling commodity c (if selling); (only appears if aa.writeUtilityComponents=true in the properties file.)
- VariationComponent; zUtility minus PriceComponent minus SizeComponent minus TransportComponent, the additional utility due to the difference between the logsum forumation (used in equation 14 and 18) and the direct average calculation for PriceComponent, SizeComponent and TransportComponent. This is the additional utility associated with having a variety of zones from which to buy or to which to sell. The total impact of commodity variety on utility is equal to the SizeComponent plus the VariationComponent – the SizeComponent measures within-zone variety while the VariationComponent measures between-zone variety.

- ConsumptionErrorTermSizes; each row concerns a specific activity and its consumption error terms, indicating the influence of the commodity error terms (for each of the commodities consumed by the activity) on the technology and location choices of the activity. This provides an indication of how the variability of each commodity (and hence also the size variables in each zone for each commodity, which are proportional to the variability) influence the sensitivity of the location and technology choice for the activity. Generally, it is important that the variability associated with the consumption of space (reported in this file) is as large as the variability associated with consuming other commodities or with producing other commodities (as reported in ProductionErrorTermSizes). Thus this file can be used in calibrating the commodity error terms through the adjustment of the BuyingDispersionParameter and SellingDispersionParameter in CommoditiesI;

  - Activity; text characters of name of activity as specified in ActivitiesI;

- *CommodityName*; numerical value of error term for commodity with text name '*CommodityName*', indicating the impact of commodity variability on activity variability.each row concerns a specific activity and its consumption error terms, indicating the values for the consumption error terms for each of the commodities consumed by the activity determined in a run of the AA Module;
  - Activity; text characters of name of activity as specified in ActivitiesI;
  - *CommodityName*; numerical value of error term for commodity with text name '*CommodityName*'.

- ExchangeResults; each row concerns a specific commodity and exchange location land use zone (LUZ), indicating the values obtained in a run of the AA Module; these can be used as the starting values in a subsequent run of the AA Module; the entire file can be used as ExchangeResultsI directly if desired, even though in general it is only the starting values for the prices, the $Price_{c,k}$ values, that are of concern;
  - Commodity; text characters of name of commodity as specified in CommoditiesI;
  - ZoneNumber; integer value of exchange location land use zone;
  - Demand; real value of the demand for the commodity in the exchange location, in units of the commodity; this is $TDem_{c,k}$ in equation 30 in the

Theoretical Formulation document;

- InternalBought; real value of internal bought quantity of commodity in the exchange location, this is the total quantity of the commodity being bought in the exchange zone by all activities internal to the model area, in units of the commodity; this is $TBInt_{c,k}$ in equation 28 in the Theoretical Formulation document;
- Exports; real value of quantity of commodity exported from exchange location; this is $Qe_{c,k}$ in equations 22 and 30 in the Theoretical Formulation document;
- Supply; real value of the demand for the commodity in the exchange location, in units of the commodity; this is $TSup_{c,k}$ in equation 31 in the Theoretical Formulation document;
- InternalSold; real value of internal sold quantity of commodity in the exchange location, this is the total quantity of the commodity being sold in the exchange zone by all activities internal to the model area, in units of the commodity; this is $TSInt_{c,k}$ in equation 29 in the Theoretical Formulation document;
- Imports; real value of quantity of commodity imported to exchange location; this is $Qi_{c,k}$ in equations 21 and 13 in the Theoretical Formulation document;
- Surplus; real value of the residual surplus quantity of supply over demand for the commodity in the exchange location, in units of the commodity; this is $Residual_{c,k}$ in equation 33 in the Theoretical Formulation document;
- Price; real value of unit exchange price for commodity in exchange location, in units of money per unit of the commodity; this is $Price_{c,k}$ in the Theoretical Formulation document;

- ExchangeResultsTotals; each row concerns a specific commodity, indicating the aggregate values (totals and averages) obtained for all exchange zones in a run of the AA Module;

- Commodity; text characters of name of commodity; this matches the commodity name used in CommoditiesI;
- Demand; real value of total demand for the commodity in all exchange locations in the model, in units of the commodity;
- InternalBought; real value of total internal bought quantity of commodity in all exchange locations, this is the total quantity of the commodity being bought in all exchange zones in the model by all activities internal to the model area, in units of the commodity;
- Exports; real value of total quantity of commodity exported from all exchange locations in the model area;
- Supply; real value of total demand for the commodity in all exchange locations in the model, in units of the commodity;
- InternalSold; real value of total internal sold quantity of commodity in all exchange locations, this is the total quantity of the commodity being sold in all exchange zones in the model by all activities internal to the model area, in units of the commodity;
- Imports; real value of total quantity of commodity imported to all exchange locations in the model area;
- RMSSurplus; real value of the square root of the sum across all model zones of the squared values of the residual surplus quantity of supply minus demand for the commodity, in units of the commodity;
- AveragePrice; real value of weighted average across all model zones of the exchange prices for the commodity, with the exchange quantities of the commodity used for the weights, in units of money per unit of the commodity;

- Histograms; each row concerns a specific commodity and histogram range as specified in the input file HistogramsI, indicating the quantity of the commodity flow with transport lengths within the range as established by the AA Module;

- Commodity; text characters of name of commodity; this matches the commodity name used in CommoditiesI;
- BuyingSelling; text character indicating whether commodity flow is for buying or selling; character 'B' indicates the commodity flow is for buying; character 'S' indicates the commodity flow is for selling. If the 'exchange regime' for the commodity is set to 'c' or 'n' then the buying flows will all be intrazonal, and the output for 'B' will not be that useful. If the 'exchange regime' for the commodity is set to 'p' or 'n' then the selling flows will all be intrazonal, and the output for 'S' will not be that useful.
- BandNumber; integer value of the histogram range for quantity;
- LowerBound; real value of lower boundary of the histogram range for quantity, in units of the interchange attribute used to define the transport lengths in the histogram as specified in the input file Histograms;
- Quantity; real value of quantity of commodity flow within histogram range, in units of the commodity;
- AverageLength; real value of the mean of the distribution of transport lengths in the histogram range.

- LatestActivityConstants; each row concerns a specific activity and location zone, indicating the components of the location utility and resulting quantity of activity allocated to each land use zone at the final iteration of the AA constraint process. This file is identical to ActivityLocations, except the LocationSpecificUtility is the zone constant that would have been used by AA in the *next* iteration of the constraint process, instead of the zone constant that was used in the *last* iteration of the constraint process. If the constraint process did not converge appropriately, this file can be used to restart it one iteration ahead, hence saving some run time.

- PctIntrazonalxCommodityxBzone; each row concerns a specific commodity and bexchange zone, indicating the quantities and proportions of the amounts bought and sold in the exchange zone by activities located in the exchange zone and in total for a run of the AA Module;

- BZone; number of LUZ; a in Equations
- Commodity; text characters of name of commodity; this matches the commodity name used in CommoditiesI;
- BuyIntra; quantity of commodity bought in zone (as an exchange zone) by activities located in zone (as a consumption zone) ; ??symbol?? in Equation ??
- BuyFrom; total quantity of commodity bought in zone (as an exchange zone) by all activities model-wide; ??symbol?? in Equation ??
- BuyTo; total quantity of commodity bought in all zones model-wide by activities in zone (as a consumption zone) ; ??symbol?? in Equation ??
- BuyPctIntraFrom; quantity of commodity bought in zone by activities located in zone as a proportion of the total quantity of commodity bought in zone by all activities model-wide ; ??symbol?? in Equation ??
- BuyPctIntraTo; quantity of commodity bought in zone by activities located in zone as a proportion of the total quantity of commodity bought in zone by all activities model-wide; ??symbol?? in Equation ??
- SellIntra; quantity of commodity sold in zone (as an exchange zone) by activities located in zone (as a production zone) ; ??symbol?? in Equation ??
- SellFrom; total quantity of commodity sold in zone (as an exchange zone) by all activities model-wide; ??symbol?? in Equation ??
- SellTo; total quantity of commodity sold in all zones model-wide by activities in zone (as a production zone) ; ??symbol?? in Equation ??
- SellPctIntraFrom; quantity of commodity sold in zone by activities located in zone as a proportion of the total quantity of commodity sold in zone by all activities model-wide ; ??symbol?? in Equation ??

  - SellPctIntraTo; quantity of commodity sold in zone by activities located in zone as a proportion of the total quantity of commodity sold in zone by all activities model-wide; ??symbol?? in Equation ??

??obviously some guesses on the definitions of these values above ?? A figure with arrows and circles is needed to help describe these too??

- ProductionErrorTermSizes; each row concerns a specific activity and its production error terms, indicating the influence of the commodity error terms (for each of the commodities produced by the activity) on the technology and location choices of the activity. This provides an indication of how the variability of each commodity (and hence also the size variables in each zone for each commodity) influence the sensitivity of the location and technology choice for the activity. Generally, it is important that the variability associated with production (reported in this file) do not overwhelm the variability associated with the consumption of space (reported in ConsumptionErrorTermSizes). Thus this file can be used in calibrating the commodity error terms through the adjustment of the BuyingDispersionParameter and SellingDispersionParameter in CommoditiesI;
  - Activity; text characters of name of activity as specified in ActivitiesI;
  - *CommodityName*; numerical value of error term for commodity with text name '*CommodityName*', indicating the impact of commodity variability on activity variability.

- Selling_*CommodityName*; origin-destination matrix of flows of commodity with text name '*CommodityName*' going from location zone (LUZ) where it is produced to exchange zone (LUZ) where it is sold; row numbers are origin (production) zones and column numbers are destination (exchange) zones; each cell in the matrix indicates the quantity of flow of the commodity in the specified units for the commodity determined in a run of the AA Module; $S_{c,z,k}$ in equation 12 in the Theoretical Formulation document.

The file may be written in zipMatrix format, which will be indicated by its filename extension; to view a file in zipMatrix format the MatrixViewer program must be run; this is done by clicking on the viewmatrix.cmd icon in the base year directory (and perhaps elsewhere) which will open a cmd window and also start the MatrixViewer program (current version 0.4); the file tab in the MatrixViewer window can then be used to open and view the file; the cell values can then be imported to an Excel spreadsheet if desired using copy (Control-C) and paste (Control-P) commands

- ZonalMakeUse; each row concerns a specific activity and commodity and location zone; for make indicating the aggregate quantity of the commodity made by the activity in the zone expressed in the specified units for the commodity together with the effective 'blended' production rate per unit of activity and the production utility per unit of the commodity produced; for use indicating the aggregate quantity of the commodity used by the activity in the zone expressed in the specified units for the commodity together with the effective 'blended' consumption rate per unit of activity and the consumption utility per unit of the commodity consumed (these expressed as negative values consistent with the use of negative technical coefficients for consumption) obtained in a run of the AA Module;
    - Activity; text characters of name of activity as specified in Activities**I**;
    - ZoneNumber; integer value of location land use zone;
    - Commodity; text characters of name of commodity; this matches the commodity name used in Commodities**I**;
    - MorU
    - Coefficient; Blended average coefficient, $TPA_{c,a,z}$ / $W_{a,z}$
    - Utility; $CUBuy_{c,z}$ or $CUSell_{c,z}$
    - Amount; $TPA_{c,a,z}$ in Equation 8

3.5 AA Module Run Control File – aa.properties

When the AA Module is run, aspects of the algorithms it is to perform and the locations of certain inputs and outputs it is to use are specified in the form of 'run controls'. These run controls are included in the 'aa.properties' file for the run, which is located in the sub-directory where the run is performed. The values for certain model parameters are also set in the 'aa.properties' file.

A 'separate' set-up for running the AA Module on its own includes a single aa.properties file in the sub-directory where the AA Module is run. A 'full' set-up for running the full model system with multiple runs of the AA Module for multiple years includes multiple aa.properties files, one in each of the sub-directories where the AA Module is run.

The run controls are specified in the aa.properties file by setting program variables equal to specific values, with one such 'equation' setting per line and the lines in any order in the file. A comment line is indicated with '#' in the first space of the line. The variables and settings are as follows:

output.data

This sets the directory path for the location where the AA Module output files are written; forward slashes are used (in place of the standard backward slashes) and the path must end with a forward slash; an example setting is:

```
output.data = C:/Pecas/58ZnAA/BaseRun/BaseOutputs/
```

where the directory path is " C:\Pecas\58ZnAA\BaseRun\BaseOutputs\ "

processor.class

This sets the version of the AA Module to use for reading the inputs and writing outputs. Custom versions of the processor can be written to interface with different databases or input file formats. There are two standard versions provided::

- for the MakeUseI version (the old version):
  processor.class = com.pb.models.pecas.PIPProcessor
- for the TechnologyOptionsI version (the new version):
  processor.class = com.pb.models.pecas.AASetupWithTechnologySubstitution

aa.useLogitTechnologyChoice

This sets whether the more general version of logit technology, as documented in the Theoretical Formulation Document should be used, or whether a simpler version should be used with implied alternatives:

- for the MakeUseI version (the old version):
  aa.useLogitTechnologyChoice = false
- for the TechnologyOptionsI version (the new version):
  aa.useLogitTechnologyChoice = true

aa.useLogitProduction

This sets the version of the AA Module to use; settings are:

- for the exponential consumption and production version (the old version) using MakeUseI,:
  aa.useLogitProduction = true
- for the TechnologyOptionsI version (the new version):
  aa.useLogitProduction = false

aa.calculateAveragePrices

This sets whether or not the AA Module is to include consideration of model-wide average prices in the price update process in the solution algorithm (using block derivatives and exact Newton Method); settings are:

- for including this consideration (recommended):
  aa.calculateAveragePrices = true
- for not including this consideration:
  aa.calculateAveragePrices = false

aa.useSQLInputs

This sets whether or not the AA Module inputs are stored in an SQL database or JDBC datasource such as Excel or Access; settings are:

- for when the inputs are stored in an SQL database or JDBC database:
  aa.useSQLInputs = true
- for when the inputs are stored in text files:
  aa.useSQLInputs = false

aa.excelInputs

This sets whether or not the AA Module inputs are stored in an Excel spreadsheet; settings are:

- for when the inputs are stored in an Excel spreadsheet file:
  aa.excelInputs = true
- for when the inputs are not stored in an SQL database table file:
  aa.excelInputs=false

```
# Input files are stored in a SQL database.

aa.useSQLInputs = true
```

```
# pointer to the input file SQL database

aa.datasource = jdbc:odbc:58AABase

aa.jdbcDriver = sun.jdbc.odbc.JdbcOdbcDriver


# input files are actually in excel -- need to tell the software this because Excel is not a real database and so needs special treatement

aa.excelInputs = true
```

aa.base.data

This sets the directory path for the location of the following input files for the AA Module:

- ActivitiesI;
- ActivitySizeTermsI;
- ActivitiesZonalValuesI
- CommoditiesI;
- ExchangeImportExportI;
- FloorspaceI,
- FloorspaceBuyingSizeTermsI;
- HistorgramsLayoutsI;
- TechnologyOptionsI;

If one of these input files (with extension .csv) is included in the indicated directory, then the values it contains are used rather than the corresponding values in the SQL database files. Forward slashes are used (in place of the standard backward slashes) and the path must end with a forward slash; an example setting is:

```
aa.base.data = C:/Pecas/58ZnAA/2014/Inputs/
```

where the directory path is " C:\Pecas\58ZnAA\2014\Inputs\ "

calculateExchangeSizes

This sets whether the buying and selling size terms for nonfloorspace commodities are to be set through an allocation with "Expected Prices". If this is set to true, an initial allocation is performed without clearing prices, with only location choice size terms and floorspace buying size terms set. This allocates activities in proportion to zone sizes. The technology choices made by activities are also uninfluenced by market conditions. The resulting quantities of commodities bought and sold are used to set the size terms for buying and selling the remaining commodities.

aa.current.data

This sets the directory path for the location of the following input files for the AA Module:

- ActivitiesW;
- ActivitiesZonalValuesW
- ActivityConstraintsI; and
- ExchangeResultsI.

If one of these input files (with extension .csv) is included in the indicated directory, then the values it contains are used rather than the corresponding values in the SQL database files. Forward slashes are used (in place of the standard backward slashes) and the path must end with a forward slash; an example setting is:

aa.current.data = C:/Pecas/58ZnAA/2014/CurrentInputs/

where the directory path is " C:\Pecas\58ZnAA\2014\CurrentInputs\ "

aa.jdbcDriver

This sets the name of the JDBC driver to be used to access the data. JDBC drivers are provided by all major database vendors, and changing this value allows PECAS to operate on data stored in various databases. Normally, PECAS uses a JDBC driver provided with Java, called the jdbc:odbc driver. This allows PECAS to access any datasource identified through the Microsoft Windows Datasources (ODBC) facility (Control Panel ->Administrative Tools->Datasources). The Windows ODBC facility provides link to Microsoft Access and Microsoft Excel, among others; To use this default driver the setting is:

```
aa.jdbcDriver = sun.jdbc.odbc.JdbcOdbcDriver
```

To use another driver, consult the documentation for your database software.

aa.datasource

This sets the pointer of the JDBC link to the database (SQL or Excel) containing the inputs for the AA Module. Once the JDBC driver is specified (see above), the JDBC driver will use this value to find the data. The format of this item depends on the JDBC driver used, if the default jdbc:odbc driver is used an example setting is:

```
aa.datasource = jdbc:odbc:58AABase
```

where ‘58AABase’ is the name given to the datasource in Control Panel->Administrative Tools->Datasources (ODBC).

aa.maxiterations

This sets the maximum number of iterations for the AA Module to perform in a run; it is the variable IMax in the Theoretical Formulation documentation; an example setting is:

```
aa.maxIterations = 300
```

The number of iterations required for the AA Module to converge is influenced by many factors, so it is not possible to provide definitive guidance on suitable values for IMax overall. That said, in general, with convergence criteria that are reasonable and not too stringent and starting prices are not too wildly different from the solution prices, it should not take more than about 500 iterations for the AA Module to converge. Numbers of iterations well beyond that without convergence may be indicative of problems making it too difficult (and perhaps even impossible) to achieve convergence.

aa.initialStepSize

This sets the size of the first step in the price update process used in the AA Module solution algorithm; it is StepInit in the Theoretical Formulation documentation; an example setting is:

```
aa.initialStepSize = 0.001
```

When the solution algorithm is making progress towards convergence, as indicated by decreases in the MClear measure of convergence with each step, the algorithm increases the size of the next step that it takes. A low value for the step size at the start, like 0.001, as shown in the example, is good – particularly in the initial stages of model development where there are larger changes being made and the starting prices are less likely to be close to the solution prices. This allows the solution algorithm to establish an appropriate search direction before moving too much at the start. For production models the initial step size is likely to be set higher, at perhaps 0.05.

aa.minimumStepSize

This sets the minimum size of step in the price update process used in the AA Module solution algorithm; it is StepMin in the Theoretical Formulation documentation; an example setting is:

```
aa.minimumStepSize = 0.01
```

When the solution algorithm is possibly not making good progress towards convergence, as indicated by increases rather than decreases in the MClear measure of convergence with each step, the algorithm reduces the size of the step that it is taking. The value set here is the minimum value, where it will stop making further reductions when encountering increases rather than decreases in MClear. In effect, the algorithm is still moving the direction the derivatives have indicated is the most appropriate, but is reducing the amount that it is moving because it has found that the result is not good when it moves a larger amount. Sometimes the solution algorithm has to get 'over a hump' before it can proceed to even lower values in the MClear measure of convergence. It will do this using the minimum step size. If this minimum step size is too small, then it will take a large number of steps to get 'over the hump', requiring large quantities of computer runtime. But the solution algorithm uses the derivatives at its current location to establish an indication of the most appropriate direction for the next step, and if the minimum step size is too large, then the solution algorithm may not be able to keep from moving beyond the range where this indication is accurate, and may then jump around 'wildly' from one step to the next. A compromise value is required, usually established by trial and re-trial for each particular model. On the basis of experience a value of 0.01, as shown in the example, is a good place to start.

If the model encounters a numerical overflow error, it will interpret the overflow error as due to a too-large step size, and may reduce the step size even lower than the minimum. Very low step sizes, well below the minimum, due to An ill-defined model.

aa.localPriceStepSizeAdjustment

This sets the factor applied to the local price adjustment calculated for each commodity in each zone as part of the price update process used in the AA Module solution algorithm; it is StepLoc in the Theoretical Formulation documentation; an example setting is:

```
aa.localPriceStepSizeAdjustment = 0.2
```

In each iteration, for each commodity, the price update process in the solution algorithm calculates a price adjustment for each zone. This combined price adjustment is the sum of (a) a local price adjustment based on the supply and demand for the commodity in the zone, multiplied by a weighting factor, and (b) an average price adjustment based on the supply and demand for the commodity model-wide. The value of the weighting factor is specified with this setting. A value of 0.2 is recommended, as shown in the example above. The combined price adjustment is then factored by the step size.

aa.maximumStepSize

This sets the maximum size of step in the price update process used in the AA Module solution algorithm; it is StepMax in the Theoretical Formulation documentation; an example setting is:

```
aa.maximumStepSize = 2.0
```

When the solution algorithm is making good progress towards convergence, as indicated by relatively large decreases in the MClear measure of convergence with each step, the algorithm increases the size of the next step that it takes. The value set here is the maximum value, where it will stop making further increases when encountering decreases rather than increases. A step size of 1.0 results in the search algorithm using Newton's Method directly in the determination of the average price adjustment, which should lead to good convergence. A somewhat larger step size may also speed convergence. But, if a much larger step size is used, and the search process moves beyond the range where the indications provided by the derivatives at its current location are accurate, then the process will again jump around 'wildly'. At that time, when it encounters an increase rather than a decrease in MClear, the algorithm will reduce the step size. An appropriate value for the maximum step size can help avoid some of this increasing and decreasing of step sizes, and thereby help speed convergence. Finding an appropriate value is again a trial and re-trial process. Experience has shown that an effective strategy for helping minimize runtimes to convergence is to start with a maximum step size of 2.0, as shown in the example, or even 2.0 divided by the value for the local price step size adjustment, and then keep reducing this value in subsequent model runs until there are comparatively few instances where the step size is reduced from the maximum.

aa.converged

This sets the value of the model-wide convergence criterion to be used in a run of the AA Module; it is TVerge in the Theoretical Formulation documentation; an example setting is:

```
aa.converged = 1e6
```

When the MClear measure of convergence reaches a value below TVerge, then the solution algorithm stops and the AA Module is deemed to have converged. Guidance on appropriate values for TVerge can be obtained by considering the values for MClear arising with specific total differences between supply and demand in the units being used. For example, if the units are dollars for all commodity values and all the commodity weights VergeWt_c are 1.0, then a combined residual of about 1000 dollars for all commodities in all zones (a comparatively small amount in most cases) would result in a value of MClear of about 1e6. Using this amount for TVerge, as shown in the example, would be requiring a solution consistent with or better than this combined residual of about 1000 dollars, a comparatively stringent requirement in most cases.

Model.skimFormat

This sets the format used for the file containing the transport conditions for movements from land use zone (LUZ) to land use zone (LUZ) (the skims) output by the TR Module (the transport model) that are input to the AA Module; settings are:

- for when the skims inputs are stored in zipmatrix format:
  Model.skimFormat = ??
- for when the skims inputs are stored in text file format:
  Model.skimFormat = ??
- for when the skims inputs are stored in TableDataSet format:
  Model.skimFormat = TableDataSet ??

aa.useFloorspaceZones

This sets whether a finer-level set of floorspace zones nested within the PECASZones are used instead of the PECASZones for particular operations performed by the AA Module, including reading in space quantities by floorspace zone and aggregating these to the PECASZones and disaggregating activity totals from the PECASZones and writing out these disaggregated activity totals by floorspace zone (to the file ActivityLocations2.csv); this ability to accept inputs and provide outputs for a finer-level set of zones can be used to facilitate the interface between the AA Module using the PECASZones (usually called the land use zones, or LUZ) and the TR Module using the floorspace zones (usually called the transport analysis zones, or TAZ); settings are:

- for when the finer-level set of floorspace zones are to be used:
  aa.useFloorspaceZones = true
- for when the finer-level set of floorspace zones are not to be used:
  aa.useFloorspaceZones = false

aa.writeUtilityComponents

This sets whether the AA Module is to output a particular set of additional values calculated internally for components of the utility values for commodities – for use in debugging; these additional values are included in extra columns in the CommodityUtilities.csv files. These can be useful in model calibration and debugging; settings are:

- for when the additional values are to be output:
  aa.writeUtilityComponents = true
- for when the additional values are <u>not</u> to be output:
  aa.writeUtilityComponents = false

aa.writeFlowMatrices

This sets whether the AA Module is to output the full set of commodity flow matrices for all commodities in the files Buying_*CommodityName* and Selling_*CommodityName*; settings are:

- for when the commodity flow matrices are to be output:
  aa.writeFlowMatrices = true
- for when the commodity flow matrices are <u>not</u> to be output:
  aa.writeFlowMatrices = false

Outputting the flow matrices can sometimes take a large proportion of the runtime for the AA Module, so if they are not required then they should not be output.

crossTabExchangeResults

This sets whether the AA Module is to output a summary of the results for the exchange zones in a file ExchangeResultsTotals.csv; settings are:

- for when the summary of the results for the exchange zones are to be output:
  crossTabExchangeResults = true
- for when the summary of the results for the exchange zones are <u>not</u> to be output:
  crossTabExchangeResults = false

skim.data

This sets the directory path for the location of the file containing the transport conditions for movements from land use zone (LUZ) to land use zone (LUZ) (the skims) output by the TR Module (the transport model) that are input to the AA Module; this indicates the source for these inputs for the AA Module to use; forward slashes are used (in place of the standard backward slashes) and the path must end with a forward slash; an example setting is:

skim.data = C:/Pecas/58ZnAA/BaseRun/Inputs/

where the directory path is " C:\Pecas\58ZnAA\BaseRun\Inputs\ "

skim.filename

This sets the name of the file containing the transport conditions for movements from land use zone (LUZ) to land use zone (LUZ) (the skims) output by the TR Module (the transport model) that are input to the AA Module; this indicates the source for these inputs for the AA Module to use; an example setting is:

skim.filename=58AABaseTime.csv

where the file is " 58AABaseTime.csv "

aa.writeExchangeDerivatives

This sets whether the AA Module is to output an additional column in the ExchangeResults.csv file that shows the partial derivative of the residual with respect to the price for each commodity in each zone, which can be used to establish target prices in zones by adjusting commodity quantities; settings are:

- for when the additional column is to be output:
  aa.writeExchangeDerivatives = true

- for when the additional column is <u>not</u> to be output:
  aa.writeExchangeDerivatives = false

aa.logFrequency

This sets the frequency for the AA Module to report to the log file certain summary information for each commodity during the solution algorithm. This frequency is expressed as the number of iterations of the solution algorithm for each report of the summary information; the summary information includes the maximum price, minimum price and largest residual in absolute magnitude across the set of zones along with the model-wide average price and model-wide total residual for each commodity; an example setting is:

aa.logFrequency = 10

which would be setting the AA Module to report the summary information every 10 iterations.

constrained

This sets whether the AA Module is to perform the activity constraints routine where it works to match specified activity quantity targets by adjusting the activity location alternative specific constants; settings are:

- for when the activity constraints routine is to be performed:
  constrained = true
- for when the activity constraints routine is <u>not</u> to be performed:
  constrained = false

constraint.iterations

This sets the (??maximum??) number of 'larger' iterations the AA Module is to perform in the activity constraints routine where it works to match specified activity quantity targets by adjusting the activity location alternative specific constants. Each larger iteration includes running the AA Module to convergence with a particular set of locations zone specific constants, comparing the resulting activity totals with the specified targets, and then calculating and applying updates to the locations zone specific constants to bring the activity totals closer to the targets; an example setting is:

constraint.iterations = 16

which would be setting the AA Module to perform (??a maximum of??) 16 'larger' iterations in the activity constraints routine.

constraint.smoothing

This sets the value for the dampening factor used when the AA Module is to perform the activity constraints routine where it works to match specified activity quantity targets by adjusting the activity location alternative specific constants. The updates to the locations zone specific constants in the activity constraints routine are multiplied by this dampening factor; an example setting is:

constraint.smoothing = 0.8

In general a value less than 1.0 for this dampening factor, such as 0.8 as shown in the example, will encourage a more steady but slower progress to a set of location zone specific constants that result in activity quantities that match the specified targets.

constraint.maxConstantChange

This sets the relative value for the maximum permitted update to a location alternative specific constant (in absolute magnitude) that can be applied in a given iteration when the AA Module is to perform the activity constraints routine where it works to match specified activity quantity targets. For each activity, the relative value set here is multiplied by the dispersion parameter for the location allocation logit model for the activity so that the value is appropriately scaled; an example setting is:

```
constraint.smoothing = 2.5
```

In general a smaller relative value for this maximum permitted update will encourage a more steady but slower progress to a set of location zone specific constants that result in activity quantities that match the specified targets. Some trial-and-retrial adjustment of the value is likely to be required, and experience has shown that a value of 2.5, as shown in the example, provides a good starting point. A warning is written to the log file by the AA Module with each instance where the calculated update is larger than the maximum and is 'clipped' to the maximum before it is applied.

3.6 Treatment of TAZ

The AA Module operates for the most part at the land use zone (LUZ) level. But a feature is available within the AA Module to include some consideration of spatial distributions at the transport analysis zone (TAZ) level, where certain inputs are accepted and certain outputs are provided at the TAZ level. This feature can be used to have the AA Module provide activity quantities at the level of the TAZ system being used by the TR Module, as part of the information flow from the AA to the TR Module. The operation of this feature is described immediately below.

The TAZ must nest within the LUZ, with a natural number of TAZ fully spanning each LUZ. The correspondence between the two zone systems is specified in the

FloorspaceZonesI input file. In general, there will be multiple numbers of TAZ within each LUZ, consistent with the use of comparatively smaller zones in transport modelling. One special case that is still possible with this feature is the one-to-one correspondence between TAZ and LUZ, where the two systems match exactly. In this special case all the consideration of spatial distributions at the TAZ level are still performed by the AA Module as described here, but the inputs and outputs for the TAZ level will match those for the LUZ level, consistent with the one-to-one correspondence between zone systems.

In each year t, quantities of space by type established in the SD Module are input to the AA Module. These quantities of space by type are determined by the SD Module and input to the AA Module at the TAZ level, using the FloorspaceI file. The AA Module aggregates these quantities of space to the LUZ level for its standard operations. These standard operations establish the quantities of activities in each LUZ for the year. The AA Module then allocates these quantities of activity in each LUZ among the TAZ nested within the LUZ according to the distributions of space by type among the TAZ and the use rates for space by type for each activity for the LUZ. The resulting quantities of activity by TAZ are then output by the AA Module and input to the TR Module using the ActivityResults file.

The equation used in the allocation of a given quantity of activity a in a given LUZ z among the TAZ nested within the LUZ, the $i \varepsilon I(z)$, is as follows:

$$TAZW_{a,i} = W_{a,z} \cdot \left(\Sigma_{c \varepsilon C(a)} \left(\left(TSpcQty_{c,i} / \Sigma_{i \varepsilon I(z)} TSpcQty_{c,i} \right) \cdot RelC_{c,a,z} \right) \right) \quad \text{for each } i \varepsilon I(z)$$

with:

$$RelC_{c,a,z} = TCA_{c,a,z} / \Sigma_{c \varepsilon C(a)} TCA_{c,a,z}$$

where:

a	= index for activity categories;

z	=	index for land use zones (LUZ);
i	=	index for transport analysis zones (TAZ);
I	=	set of all transport analysis zones;
I(z)	=	set of all transport analysis zones (TAZ) in land use zone (LUZ) z; *in software: listed in FloorspaceI;*
c	=	index for space types (treated as commodities in the AA Module); *in software: update*
C(a)	=	set of all space types used by activity a;
$TAZW_{a,i}$	=	quantity of activity a in transport analysis zone (TAZ) i; *in software: ActivityLocations(Quantity) when the feature to consider TAZ is used*
$W_{a,z}$	=	quantity of activity a in land use zone (LUZ) z;
$TSpcQty_{c,i}$	=	quantity of space type c in transport analysis zone (TAZ) i; *in software: update;*
$TCA_{c,a,z}$	=	quantity of commodity c (in this case space type c) consumed by activity a in zone z using all technology options; *in software: ZonalMakeUse(Negative of Amount when Activity=a, MorU=M, ZoneNumber=z)*
$RelC_{c,a,z}$	=	proportion of total space consumption by activity a in zone z that is consumption of space type c – indicating the relative importance of space type c compared to other space types for activity a in zone z.

This allocation to the TAZ is performed for each activity in each LUZ, thereby establishing the full set of $TAZW_{a,i}$ values for all aεA and iεI for the year.

This allocation process preserves the value established by the AA Module for the land use zone for the proportion of the total physical quantity of each space type that is available as a commodity and consumed in the land use zone, $SpcPrp_{c=h,z}$, in equation 26 in the Theoretical Formulation document. That is, it establishes for each space type an allocation among the transport analysis zones with the same proportion in each transport analysis zone as the proportion in the land use zone.

The AA Module is instructed to use this feature to accept certain inputs and provide certain outputs at the TAZ level by setting the 'aa.useFloorspaceZones' program variable in the aa.properties file to 'true', which is done using the statement 'aa.useFloorspaceZones = true'.

3.7 Treatment of Size Terms

A size term is included in the utility function used by the AA Module in the allocation of activity quantities to zones. This size term, $Size_{a,z}$ in equation 02 in the Theoretical Formulation document, provides representation of the relative size of the zone for the activity, indicating the *a priori* expected share of the activity for the zone out of the full set of zones in the model.

There are two basic approaches available with the AA Module for establishing the values to be used for the size term.

Exogenous Sizes Approach to Specification of Size Term Values

One approach, called the ‘exogenous sizes’ approach, is to calculate the values for the size terms exogenously and then input directly into the AA Module. The values are input using the ActivitiesZonalValuesI input file, with the value for the variable SizeTerm listed for each activity for each zone. A different set of values can be specified for each year considered with the AA Module.

Any combination of activity and zone not listed in ActivitiesZonalValuesI is assigned a size term value of 1 by default, unless it is assigned a size term value using the other approach described immediately below, which takes precedence.

Endogenous Sizes Approach to Specification of Size Term Values

The other approach, called the ‘endogenous sizes’ approach, is to specify how the AA Module is to build-up the values endogenously using appropriate space quantities determined by the model. This allows the values to change from one year to the next in response to changes in space quantities determined by the model.

With the endogenous sizes approach, the relative size of the zone is calculated using the values of a weighted sum of the quantities of space of different types in the zones. For a given activity, the weighted sum of space quantity for each zone is calculated as follows:

$SpSSum_{a,z} = \Sigma_{c\varepsilon C(a)} SpSWgt_{c,a} \cdot SpcQty_{c,z}$

where:

a	=	index representing activity category;
z	=	index representing land use zones (LUZ);
c	=	index representing space types, which are represented as certain commodity types in the AA Module;
C(a)	=	set of space types included in definition of weighted sum of floorspace quantity for activity category a as specified in ActivitySizeTermsI; *in software: indicated by listing in ActivitySizeTermsI ;*
$SpSSum_{a,z}$	=	weighted sum of space quantity for activity category a in zone z;
$SpSWgt_{c,a}$	=	weight specified for space type c for the calculation of the weighted sum of space quantity for activity category a; *in software: ActivitySizeTermsI(Weight);*
$SpcQty_{h,i}$	=	quantity of space of type c in zone z; *in software: FloorspaceI(Quantity);*

The proportional values used for the corresponding size terms, $Size_{a,z}$ in equation 02 in the Theoretical Formulation document, are then calculated as follows:

$Size_{a,z'} = SpSSum_{a,z'}$ for each $a\varepsilon A(s)$ and $z'\varepsilon Z$

where:

A(s)	=	set of activity categories for which weighted sum of space quantities are used to establish size terms for allocation to zones as specified in ActivitySizeTermsI; *in software: indicated by listing in ActivitySizeTermsI ;*
z’	=	index representing specific land use zones (LUZ);
Z	=	set of land use zones (LUZ) in the model.

is instructed to use this feature to accept certain inputs and provide certain outputs at the TAZ level by setting the ‘aa.useFloorspaceZones’ program variable in the aa.properties file to 'true', which is done using the statement 'aa.useFloorspaceZones = true'.

When there are conflicts in the specification of size terms, where a particular activity and zone are considered in both the 'exogenous sizes' and 'endogenous sizes' approaches, the specification in the 'endogenous sizes' approach takes precedence. Any combination of activity and zone not considered in either approach is assigned a size term value of 1 by default. If none of the zones for an activity are considered in either approach, and thus all of the zones for the activity are assigned a size term value of 1, this effectively removes the influence of the size term in the allocation of the activity to zones.

3.8 Managing and Monitoring the Solution Algorithm

The AA Module searches for the set of prices that establishes the equilibrium solution where all the markets clear – where the total demand minus the total supply minus the total demand for each commodity c in each exchange zone k, denoted $\text{Residual}_{c,k}$, is 0. It performs this search in a series of iterations.

In each iteration a new set of prices is considered. The extent that the values for $\text{Residual}_{c,k}$ are not 0 with these prices is evaluated and appropriate updates to these prices are calculated and applied. The process then moves to the next iteration with these prices used as the new set of prices.

This search process is managed by specifying the criteria that must be satisfied in order for the process to stop and by controlling elements of the calculation of the updates to prices in each iteration. These elements of this management are outlined below.

Measuring Extent of Convergence and Specifying Stopping Rules

In each iteration, the search process measures the extent that the values for $\text{Residual}_{c,k}$ are not 0 using a weighted sum-of-squares value as follows:

$$\text{MClear} = \Sigma_{c\varepsilon C}\, \text{VergeWt}_c^2\, \Sigma_{k\varepsilon K}\, (\text{Residual}_{c,k})^2 \tag{35}$$

where:

MClear	=	residual-squared measure of the extent that all markets have not cleared and the condition that $\text{Residual}_{c,k} = 0$ for all c and k has not been satisfied; a value of 0 indicates the condition has been fully satisfied *in software: GOF output to logfile ??names of these??*
VergeWt_c	=	weight applied to residual-squared values for commodity c to account for units effects when summing residual-squared values across commodities; *in software: CommoditiesI(GOFWeighting).*

The search process uses this value in its determination of the appropriate update to the prices for the next iteration.

The search process reports the value of MClear in each iteration, by writing it to the log file. This value is influenced by the units used for the commodities and by the size of the markets considered, which can make it difficult to interpret. A normalized form of the measure is also calculated and reported as follows:

$$\text{TClear} = [\ \Sigma_{c\varepsilon C}\, \text{VergeWt}_c^2\, \Sigma_{k\varepsilon K}\, (\text{Residual}_{c,k})^2\ ]^{½} /\ \text{AveExchgTotal} \tag{36}$$

with:

$$\text{AveExchgTotal} = [\ \Sigma_{c\varepsilon C}\, \text{VergeWt}_c^2\, \Sigma_{k\varepsilon K}\, (0.5^*(\text{TSup}_{c,k} + \text{TDem}_{c,k}))^2\ ]^{½} \tag{37}$$

where:

TClear = normalized measure of the extent that all markets have not cleared and the condition that $Residual_{c,k} = 0$ for all c and k has not been satisfied; a value of 0 indicates the condition has been fully satisfied
in software: GOF output to logfile ??names of these??

TClear may be much larger than 1 in the initial iterations of a search, but will quickly drop to much smaller values on its way to a value of 0 when the search process is converging to a solution.

The search process is terminated when either (a) the values for $Residual_{c,k}$ satisfy specified convergence criteria indicating they are sufficiently close to 0, or (b) the specified maximum number of iterations is reached.

The maximum number of iterations, Imax, is specified in the aa.properties Run Control file using the setting for aa.maxiterations .

The number of iterations required for the AA Module to converge is influenced by many factors, so it is not possible to provide definitive guidance on suitable values for IMax overall. That said, in general, with convergence criteria that are reasonable and not too stringent and starting prices are not too wildly different from the solution prices, it should not take more than about 500 iterations for the AA Module to converge. Numbers of iterations well beyond that without convergence may be indicative of problems making it too difficult (and perhaps even impossible) to achieve convergence.
Two basic types of convergence criteria regarding the values for $Residual_{c,k}$ are included. Both use normalized measures that are always positive and are 0 at the equilibrium solution. One concerns the extent that all markets are cleared, the other concerns the extent that individual markets are cleared. Each is described below:

(a) Total Clearance Criterion:

The 'Total Clearance' criterion concerns the extent of market clearance for all commodities in all markets altogether, and uses the TClear measure defined above. A maximum value is specified and TClear must be less that this maximum value in order for the criterion to be satisfied. This maximum value is denoted MaxTClear, and it is specified in the aa.properties Run Control file using the setting for aa.maxTotalClearance

In general, values of MaxTClear in the range of 0.001 are used.

Further guidance on appropriate values for MaxTClear can be obtained by considering the values for MClear and AveExchgTotal arising in a typical economy with specific total differences between supply and demand in the units being used. For example, if the units are dollars for all commodity values and all the commodity weights VergeWt_c are 1.0, then a combined residual of about 1000 dollars for all commodities in all zones (a comparatively small amount in most cases) would result in a value of MClear of about 1e6. With a total economy of 10 billion dollars of exchanges, the value of AveExchgTotal is 1e10, so the resulting value of TClear is 1e-7 .

The search algorithm reports the value of TClear in each iteration, making it possible to monitor the progress towards satisfying the requirement that TClear be less than MaxTClear.

(b) 'Specific Clearance' Criterion:

The 'Specific Clearance' criterion concerns the extent of clearance for specific commodities in individual markets. It uses a measure for an individual commodity and market as follows

$$\text{SClear}_{c,k} = | \text{Residual}_{c,k} | / \text{SingleExchgTotal}_{c,k} \qquad (38)$$

with:

$$SingleExchgTotal_{c,k} = \left| (0.5*(TSup_{c,k} + TDem_{c,k})) \right| + ConFac \cdot AveExchgTotal / VergeWt_{c} \qquad (39)$$

where:

ConFac	=	factor adjusting the scaled contribution to the denominator ensuring it is not zero in $SClear_{c,k}$ *in software: GOF output to logfile ??names of these??*

A maximum value is specified and all $SClear_{c,k}$ for all c and k must be less than this maximum value in order for the criterion to be satisfied. This maximum value is denoted MaxSClear, and it is specified in the aa.properties Run Control file using the setting for aa.maxSpecificClearance .

In general, values of MaxSClear in the range of 0.001 are used.

The value for ConFac is specified in the aa.properties Run Control file using the setting for aa.contributionfactorscale. In general, a value approximately equal to the reciprocal of the number of LUZ in the model is used.

Certain SClear values may be much larger than 1 in the initial iterations of a model run, but will quickly drop to much smaller values on their way to values of 0 when the model is converging to a solution.

The search algorithm reports the maximum value of $SClear_{c,k}$ for each commodity c across the set of exchange zones k in each iteration, making it possible to monitor the progress towards satisfying the requirement that all the $SClear_{c,k}$ be less than MaxSClear.

Controlling Price Update Calculation

The updates to the prices in a given iteration are calculated seeking to minimize the value of MClear.

These updates to the prices are calculated by first calculating initial 'full update' values that are then multiplied by a step size value in order to get the values that are applied in the iteration.

The initial 'full update' in the price for each commodity in each exchange zone is the sum of (a) an adjustment calculated for an average price across all exchange zones and (b) an adjustment calculated for the local price for just that exchange zone multiplied by the a weighting factor called the Local Step Size Adjustment factor, or StepLoc.

The value for StepLoc used in this calculation is specified in the aa.properties Run Control file using the setting for aa.localPriceStepSizeAdjustment. It can be used to adjust the relative contribution of average price change using an exact Newton's method and the local price change using a local derivative.

In general, a value of 0.2 is used for StepLoc.

The initial 'full update' values obtained as described above are then multiplied by the Full Step Size Adjustment factor, Step, in order to get the 'adjusted update' values used in the iteration.

If these adjusted update values result in a lower (better) value for MClear, then the software automatically increases the value of Step slightly for the next iteration. Further such increases in Step will be made, if appropriate, in subsequent iterations until a specified maximum value for Step is reached. The search process will continue with this maximum value for Step being used as long as lower values for MClear are being obtained. If these adjusted update values results in a higher (poorer) value for MClear, then these values are not applied and the value of Step is reduced, and a new and smaller value of Step is used to calculate new adjusted update values to replace the abandoned ones. Further reductions to Step will be made, if required, until it reaches a specified minimum value. Iterations will continue with this minimum value until lower values for MClear are being obtained.

The initial, maximum and minimum values for Step – StepInit, StepMax and StepMin, respectively – are specified in the aa.properties Run Control file using the settings for aa.initialStepSize, aa.maximumStepSize and aa.minimumStepSize, respectively.

Regardless of these settings, if the current value of Step results in a numerical overflow, the value of Step is reduced even if doing so would result in it being below the specified minimum value.

A low value for StepInit, around 0.001, is appropriate, particularly in the initial stages of model development where there are larger changes being made and the starting prices are less likely to be close to the solution prices. This allows the solution algorithm to establish an appropriate search direction before moving too much at the start.

The value for StepMin is the minimum value of Step the search process can use when encountering increases rather than decreases in MClear. When the search process has been forced to use StepMin, it is still moving the direction the derivatives have indicated is the most appropriate, but is reducing the amount that it is moving because it has found that the result is not good when it moves a larger amount. Sometimes the search process has to get 'over a hump' before it can proceed to even lower values in the

MClear measure of convergence. It will do this using the minimum value for Step. If this minimum value is too small, then it will take a large number of iterations to get 'over the hump', requiring large quantities of computer runtime. But the search process uses the derivatives at its current location to establish an indication of the most appropriate direction for the next iteration, and if the minimum value for Step is too large, then the solution algorithm may not be able to keep from moving beyond the range where this indication is accurate, and may then jump around 'wildly' from one iteration to the next. A compromise value is required, usually established by trial and re-trial for each particular model. On the basis of experience a value of 0.01 for StepMin is a good place to start.

The value for StepMax is the maximum value of Step the search process can use when encountering decreases in MClear. When the search process is using StepMax, it is making consistently good progress towards convergence. A value of 1.0 for Step results in the search algorithm using Newton's Method directly in the determination of the average price adjustment, which should lead to good convergence. A somewhat larger value for Step may further speed convergence. But, if a much larger value for Step is used, and the search process moves beyond the range where the indications provided by the derivatives at its current location are accurate, then the process may jump around 'wildly'. At that time, when it encounters an increase rather than a decrease in MClear, the algorithm will reduce the value of Step. An appropriate maximum value for Step can help avoid some of this increasing and decreasing of Step, and thereby help speed convergence. Finding an appropriate value is again a trial and re-trial process. Experience has shown that an effective strategy for helping minimize runtimes to convergence is to start with a value of 2.0 for the maximum value of Step, or even 2.0 divided by the value for the local price step size adjustment, and then keep reducing this value in subsequent model runs until there are comparatively few instances where Step is reduced from the maximum value.

3.9 Activity Constraints Routine

The AA Module software includes a routine for constraining the AA Module to match specified target values for activity quantities in land use zones. This routine adjusts the zonal alternative specific constants for activity location, $\text{Constant}_{a,z}$ in equation 02 in the Theoretical Formulation document, in order to get the activity quantities to match targets specified for certain zones, up to and including all zones.

These adjustments are made in a number of 'larger' iterations. Each of these larger iterations includes running the AA Module to convergence with a particular set of locations zone specific constants, comparing the resulting activity totals with the specified targets, and then calculating and applying updates to the locations zone specific constants to bring the activity totals closer to the targets. The update equation for the zonal alternative specific constant for a given activity in a given zone is as follows:

$$\text{NuConstant}_{a,z} = \text{CrConstant}_{a,z} + \text{UpConstant}_{a,z}$$

with:

$$\text{UpConstant}_{a,z} = \text{UpFac} \cdot 1/\lambda_{l,a} \cdot \ln [\ \text{TargetW}_{a,z} / W_{a,z}\ ]$$

and further constrained where :

if $\text{UpConstant}_{a,z} > \text{UpMax} \cdot \lambda_{l,a}$ then $\text{UpConstant}_{a,z} = \text{UpMax} \cdot \lambda_{l,a}$

or

if $\text{UpConstant}_{a,z} < -1 \cdot \text{UpMax} \cdot \lambda_{l,a}$ then $\text{UpConstant}_{a,z} = -1 \cdot \text{UpMax} \cdot \lambda_{l,a}$

where:

$\text{NuConstant}_{a,z}$ = new value of $\text{Constant}_{a,z}$ for use in next 'larger' iteration;
$\text{CrConstant}_{a,z}$ = value of $\text{Constant}_{a,z}$ used in current 'larger' iteration;

$UpConstant_{a,z}$	=	calculated update to $Constant_{a,z}$ arising from results of current 'larger' iteration concerning mis-match between obtained value, $W_{a,z}$, and specified target value, $TargetW_{a,z}$, for quantity of activity a in zone z;
UpFac	=	dampening factor adjusting calculated update values; *in software: aa.properties(constraint.smoothing);*
$TargetW_{a,z}$	=	specified target value for quantity of activity a in zone z; *in software: ActivityConstraintsI(Quantity);*
$W_{a,z}$	=	quantity of activity a in zone z; *in software: ActivityLocations(Quantity);*
$\lambda_{l,a}$	=	utility function dispersion parameter for allocation of activity a among location zones; *in software: ActivitiesI(LocationDispersionParameter);*
UpMax	=	maximum permitted unscaled update value; *in software: aa.properties(constraint.maxConstantChange);*

The larger iterations are continued until either (a) the values for the updates to the constants, $UpConstant_{a,z}$, satisfy a specified convergence criterion indicating they are sufficiently close to 0, or (b) the specified maximum number of larger iterations is reached.

The maximum number of larger iterations, IConMax, is specified in the aa.properties Run Control file using the setting for aa.max.constraint.iterations .

The number of larger iterations required for the AA Module to converge in this routine is influenced by many factors, so it is not possible to provide definitive guidance on suitable values for IConMax overall. That said, in general, with convergence criteria that are reasonable and not too stringent and starting prices are not too wildly different from the solution prices, it should not take more than about 30 larger iterations for the AA Module to converge in this routine. Numbers of iterations well beyond that without convergence may be indicative of problems making it too difficult (and perhaps even impossible) to achieve convergence.

The convergence criterion for the activity constraints routine concerns the extent that the allocated values match the corresponding target values, this expressed in terms of the size of the calculated updates to the associated activity location utility constants, the $UpConstant_{a,z}$. It uses a measure for an individual activity and location zone as follows

$$ADiff_{a,z} = | UpConstant_{a,z} | / AdjW_{a,z} \tag{38}$$

with:

$$AdjW_{a,z} = | W_{a,z} | + ConFac \cdot TW_a \tag{39}$$

where:

ConFac = factor adjusting the scaled contribution to the denominator ensuring it is not zero in $ADiff_{a,z}$
in software: GOF output to logfile ??names of these??

The value of $ADiff_{a,z}$ approaches 0 as the allocated quantity of activity a in zone z approaches the specified target value.

A maximum value is specified and all $ADiff_{a,z}$ for all a and z must be less than this maximum value in order for the criterion to be satisfied. This maximum value is denoted MaxADiff, and it is specified in the aa.properties Run Control file using the setting for constraint.max.activity.difference .

In general, values of MaxADiff in the range of 0.001 are used.

The value for ConFac is specified in the aa.properties Run Control file using the setting for aa.contributionfactorscale. In general, a value approximately equal to the reciprocal of the number of LUZ in the model is used.

The AA Module is instructed to run this routine constraining it to match specified zonal activity targets by setting the 'constrained' program variable in the aa.properties file to 'true', which is done using the statement 'constrained=true'. The values for UpFac, UpMax and IConMax are specified in the aa.properties Run Control file using the settings for constraint.smoothing, constraint.maxConstantChange, and max.constraint.iterations, respectively. The target values for the zonal activity quantities, $\text{TargetW}_{a,z}$, are specified in ActivityContraintsI.

The values obtained for the $\text{Constant}_{a,z}$ after the 'larger' iterations have been completed – that incorporate the updates made in each of these 'larger' iterations and result in the $W_{a,z}$ at the market clearing solution established by the AA Module – are output as the LocationSpecificUtility values in the ActivityLocations.csv file.

There are two potential applications for this activity constraints routine. One application is in the calibration of the AA Module – establishing values for the $\text{Constant}_{a,z}$ that get the AA Module to match specified calibration targets for activity quantities. This application provides one of the tools available for model calibration as discussed in the Model Development document. The other application is in the use of the model to analyze the impacts of specific constraints on activity quantities in zones. These specific constraints can be physical or legal, or can be based on forecasts, with the analysis considering the impacts of different forecast values for some quantities on other elements of the system.

Inconsistencies in the target values will adversely affect the ability of the routine to match these targets. For example, if the model-wide sum of the specified target values for a given activity does not agree with the total quantity of the activity input to the AA Module, then the routine will continue to adjust the constants without resolution.

4. SD Module Architecture

4.1 SD Module Software Code

The SD Module is written in Java.

The SD Module runs in close association with a database containing the attributes for the full set of parcels included in the model system, called the 'parcel attribute database' or 'PA database'. One of several different commercially available database software packages can be used to implement and then to access and manipulate the PA database. It is highly desirable (but not absolutely essential) for this database software to have GIS capabilities. Certain macros and GUIs that work with this database system are also included to perform routines required as part of the interaction with the SD Module.

Database software packages that have been used in previous installations include:

Microsoft Access, alone and in combination with ArcGIS 9;
SQLServer;
SQLite; and
MySQL.

Certain macro-language scripts for running automated processes with these packages, for linking with the SD Module and with GIS and GUI operations, have been developed and may be available for use in future installations.

4.2 External Information Flows For SD Module

Figure 3 shows the information flows in and out of the SD Module and the associated GIS and GUIs in the step from a given year t to year t+1 in the full set-up. It indicates the origins and destinations of these information flows and also upper-case letter codes that are used to identify specific files and variables included in these flows. Table 3 provides a summary of these flows and their components using the letter codes in Figure 3.

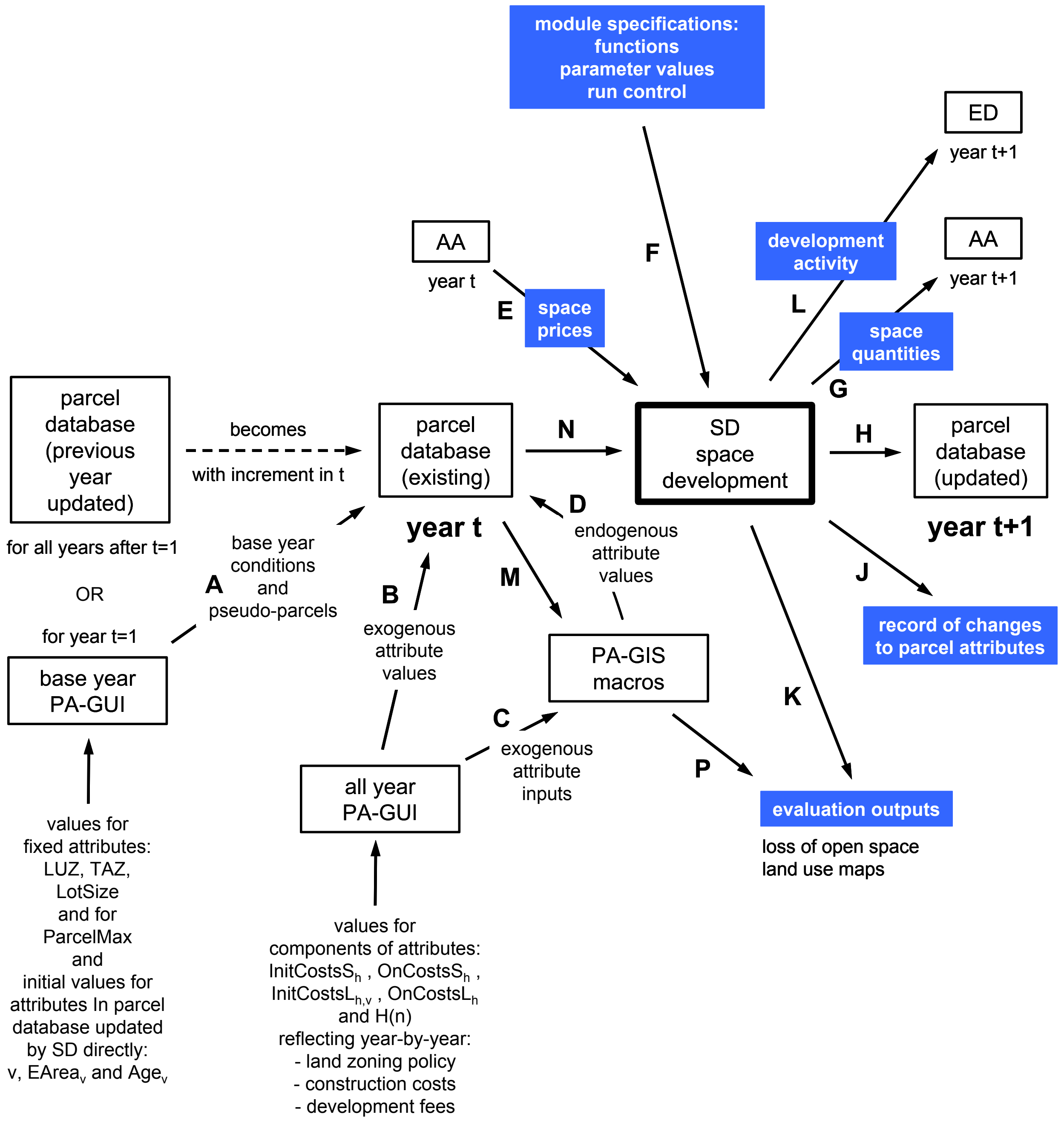

module specifications: functions parameter values run control
ED
year t+1
AA
year t+1
AA
year t
F
development activity
L
space prices
E
space quantities
G
parcel database (previous year updated)
becomes
with increment in t
parcel database (existing)
N
SD space development
H
parcel database (updated)
year t
D
endogenous attribute values
year t+1
for all years after t=1
OR
for year t=1
A
base year conditions and pseudo-parcels
B
exogenous attribute values
M
J
record of changes to parcel attributes
base year PA-GUI
PA-GIS macros
K
C
exogenous attribute inputs
P
all year PA-GUI
evaluation outputs
loss of open space
land use maps
values for fixed attributes: LUZ, TAZ, LotSize and for ParcelMax and initial values for attributes In parcel database updated by SD directly: v, EArea$_v$ and Age$_v$
values for components of attributes: InitCostsS$_h$, OnCostsS$_h$, InitCostsL$_{h,v}$, OnCostsL$_h$ and H(n) reflecting year-by-year:
- land zoning policy
- construction costs
- development fees

Figure 3: Information flows in and out of the SD Module and associated GUI and GIS Macro components in the step from a year t to a year t+1 put in the 'full' set-up where the SD Module is run in each year-to-year step in combination with the other system modules. Tables 3 and 4 list the components of these information flows using the letter codes (A through L) shown.

Table 3: Component files and variables in SD Module information flows in Figure 2

Flow in Figure 3	File	Contents
F	DevelopmentTypesI	
F	LocalLevelEffectsI	
F	ZoningSchemesI	Definitions of
F	ServiceCostCodesI	Definitions of codes for contributions to
F	TransitionConstantsI	
N	??specific filename now for working parcel database?? - before SD Module is run	set of parcel-level attributes – as listed in Table 4 below
H	??specific filename now for working parcel database?? - after SD Module is run	set of parcel-level attributes – as listed in Table 4 below
K	??specific names of files Dimantha used to consider aggregate transitions in Baltimore model??	Matrix showing quantities of space transitions and related land transitions by space type, TAZ and LUZ
P	??any such files??	Maps showing space type and quantity at parcel level

C	ExchangeResultsTotals	Summary of model-wide values for prices and exchange quantities obtained by commodity
E	ExchangeResults	Values for prices and exchange quantities for commodities, including prices for space by type and LUZ in particular
G	Floorspace	Space quantities by type and LUZ output by space development and to be used for activity allocation
L	SpaceConstuction Note: not yet included	Measures indicating quantities of construction activity arising with space development
J	SpaceChanges ??JEA: what is current filename for this??	parcel-level list of changes to space quantities

Table 4: Parcel-level attributes in SD Module information flows in Figure 2

Flow in Figure 3 Take out Later	**Attribute Name**	**Description**	**Flow(s) in Figure 3 where attribute included**
	ID	unique ID for each parcel	A, B, C, D, H, M, N, P (all flows of parcel attributes)
	LUZ	land use zone containing parcel	A, H, N
	TAZ	transport analysis zone containing parcel	A, H, N
	LotSize	land area of parcel	A, H, N
	v	initial existing space type, v, for each parcel for base year	A
	$EArea_v$	initial existing space quantity, $EArea_v$, for each parcel for base year	A
	Age_v	initial existing space age, Age_v, for each parcel for base year	A
	v	existing space type for each parcel for each year after base year – this is updated to reflect updated space type established in previous year run of SD Module	N, and H as updated
	$EArea_v$	existing space quantity for each	N, and H as updated

	parcel for base year – this is updated to reflect updated space quantity established in previous year run of SD Module	
Age_v	existing space age for each parcel for base year – this is updated to reflect 'no change' transitions established in previous year run of SD Module	N, and H as updated
E0 permitted?	indication whether developer event E0 is permitted for each parcel for each year	B
Er permitted?	indication whether developer event Er is permitted for each parcel for each year	B
Ed permitted?	indication whether developer event Ed is permitted for each parcel for each year	B
H(n)	set of new space types permitted (with developer event En) for each parcel for each year	B
$MinArea_h$	minimum area for each permitted new space type for each parcel for each year	B
$MaxArea_h$	maximum area for each permitted new space type for	B

	each parcel for each year
B	principal unit costs for transitioning to each available updated space type per unit of space, InitCostsS$_h$, for each parcel for each year
B	ongoing unit costs for transitioning to each available updated space type per unit of space, OnCostsS$_h$, for each parcel for each year
B	ongoing unit costs for transitioning to each available updated space type per unit of land, OnCostsL$_h$, for each parcel for each year
D	principal unit costs for transitioning to each available updated space type per unit of land, InitCostsL$_{h,v}$, for each parcel for each year
D	density in local area, Den, for each parcel for each year
D	distance from each local-level effect, Dist$_g$ for each parcel for each year

4.3 SD Module Run Control File

--ld.properties ??DDS check and correct and expand as required – so it is like the treatment for aa.properties (see section 4.6 above) ??

When the SD Module is run, aspects of the algorithms it is to perform and the locations of certain inputs and outputs it is to use are specified in the form of 'run controls'. These run controls are included in the 'ld.properties' file for the run, which is located in the sub-directory where the run is initiated. The values for certain model parameters are also set in the 'ld.properties' file.

A set-up for running the full model system with multiple runs of the SD Module for multiple years includes multiple ld.properties files, one in each of the directories where the SD Module is run.

The run controls are specified in the ld.properties file by setting program variables equal to specific values, with one such 'equation' setting per line and the lines in any order in the file. A comment line is indicated with '#' in the first space of the line. The variables and settings are as follows:

LandJDBCDriver ??JEA: check this??

This does ???; an example setting is:

```
LandJDBCDriver=org.sqlite.JDBC
```

InputJDBCDriver=sun.jdbc.odbc.JdbcOdbcDriver

```
InputJDBCDriver=sun.jdbc.odbc.JdbcOdbcDriver
```

LandDatabase

```
LandDatabase=jdbc:sqlite:C:/BaltimorePECAS2005/SQLiteTest/sqliteparcels.db
```

InputDatabase=jdbc:odbc:BaltimorePECAS

```
InputDatabase=jdbc:odbc:BaltimorePECAS
```

ExcelInputDatabase

```
ExcelInputDatabase=true
```

ExcelLandDatabase

```
ExcelLandDatabase=false
```

LandTable

```
LandTable=parcels
```

UseSQLInputs

```
UseSQLInputs=true
```

UseSQLParcels

```
UseSQLParcels=true
```

DevelopmentDispersionParameter

```
DevelopmentDispersionParameter=.0000005
```

DevelopmentAlternativesDispersionParameter

DevelopmentAlternativesDispersionParameter=0.0000005

IntensityDispersionParameter

This sets the value for the utility function dispersion parameter for the allocation of updated space quantities, λ_{q,h^*} in equations 67 through 69 in the Theoretical Formulation document; an example setting is:

IntensityDispersionParameter=.000001

Note that the same single value is specified for all space types h.

LUZColumnName

This sets the label used in the parcel database for the land use zone (LUZ) attribute for the parcel (indicating the LUZ in which the parcel is located) in each parcel record; an example setting is:

LUZColumnName=ID3

where the label is 'ID3'

TAZColumnName

This sets the label used in the parcel database for the transport analysis zone (TAZ) attribute for the parcel (indicating the TAZ in which the parcel is located) in each parcel record; an example setting is:

TAZColumnName=ID2

where the label is 'ID2'

CoverageColumnName

This sets the label used in the parcel database for the space type attribute for the parcel (indicating the type of space developed on the parcel) in each parcel record; an example setting is:

```
CoverageColumnName=DevelopmentTypeCode
```

where the label is ‘DevelopmentTypeCode’

QuantityColumnName

This sets the label used in the parcel database for the space quantity attribute for the parcel (indicating the quantity of space developed on the parcel) in each parcel record; an example setting is:

```
QuantityColumnName = AmountOfDevelopment
```

where the label is ‘AmountOfDevelopment’

ZoningColumnName

```
ZoningColumnName=ZoningSchemeCode
```

This sets the label used in the parcel database for the zoning scheme attribute code used to identify the specific set of zoning rules for the parcel in each parcel record; an example setting is:

```
ZoningColumnName = ZoningSchemeCode
```

where the label is ‘ZoningSchemeCode’

SizeColumnName

This sets the label used in the parcel database for the quantity of land attribute for the parcel in each parcel record; an example setting is:

SizeColumnName = AmountOfLand

where the label is ‘AmountOfLand’

YearBuiltColumnName

This sets the label used in the parcel database for the year built attribute for the space on the parcel in each parcel record; an example setting is:

YearBuiltColumnName = YearBuilt

where the label is ‘YearBuilt’

AAResultsDirectory

AAResultsDirectory = C:/BaltimorePECAS2005/Baltimore PECAS InAndOut/

LogFilePath

LogFilePath = C:/BaltimorePECAS2005/Baltimore PECAS InAndOut/

IntegerCodeForCoverage

IntegerCodeForCoverage = true

MaxParcelSize

MaxParcelSize = 1

InputParcelFile

InputParcelFile = C:\\BaltimorePECAS2005\\SQLiteTest\\sqlitebulkload.sql

OutputParcelFile

This sets the directory path and filename for the file where the SD Module outputs are written; forward slashes are used (in place of the standard backward slashes); an example setting is:

OutputParcelFile = C:/Pecas/BaseRun/BaseOutputs/SDResults.csv

where the directory path is " C:\Pecas\BaseRun\BaseOutputs\SDResults.csv "

4.4 SD Module Input Files and Variables

The structure and contents of the input files for the SD Module are indicated below.

- DevelopmentTypesI Each row concerns a specific space type, indicating the name and parameter values for the space type:
  - DevelopmentTypeName; text characters of name of space type; this must match exactly the commodity name used for this space type in its representation as a commodity in the AA Module;
  - GridCodeInt; integer value of index for space type, used for both v and h indices for existing and updated space types;
  - Units; text characters of units used for expressing quantities of space type; this must match exactly the units designation for the commodity corresponding to this space type as specified in the CommoditiesI input file for the AA Module;
  - ConstructionCost; real value of unit cost for construction of new space type, for the En development event, in units of money per unit of space type; this value is used to calculate the $TrCostsS_{v,h}$ and $TrCostsL_{v,h}$ within

the SD Module in the previous version of the SD Module; in the current version of the SD Module the construction cost inputs to $TrCostsS_{v,h}$ and $TrCostsL_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;

- ReconstructCost; real value of unit cost for construction of altered or renovated space type, for the Er development event, in units of money per unit of space type; this value is used to calculate the $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the reconstruction cost inputs to $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- MaintenanceCost; real value of base unit costs for maintaining space type (when it has an age of 1 year); expressed as an amount per unit of space per year; this is $MtBase_h$ in equation 75 in the Theoretical Formulation document;
- AddASC; real value of transition constant for 'alter or renovate' development event Er for the space type; this is $TrConst_{v,h=vEr}$ in the Theoretical Formulation document;
- SameASC; real value of transition constant for 'no change' development event E0 for the space type; this is $TrConst_{v,h=vE0}$ in equation 55 in the Theoretical Formulation document;
- AgeRentDiscount; real value of proportional decrease in rent for space type with each additional year of age; expressed as a proportion without units; this is $AgeFac_h$ in equation 70 in the Theoretical Formulation document;,
- DensityRentDiscount; real value of proportional decrease in rent for space type with each additional increase in density in the area around the parcel; expressed as a proportion without units; this is $DenFac_h$ in equation 70 in

the Theoretical Formulation document;,

- RegPriceEffect; ??update??
- RegPriceReference; ??update??
- LandRent; ??update??
- TypicalFAR; ??update??
- AgeMaintenanceCost; real value of proportional increase in maintenance costs for space type with each additional year of age; expressed as a proportion without units; this is $MtFac_h$ in equation 76 in the Theoretical Formulation document;
- DemolitionCost; real value of unit cost for demolition of existing space type, in units of money per unit of space type; this value is used to calculate the $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the demolition cost inputs to $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- ServiceRequirement; real value of rate at which services are required for space type, in units of money per unit of space type; this value is used to calculate the $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the servicing cost inputs to $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- VacantType; ??update??

- LocalLevelEffectsI; each row concerns a specific local-level effect on the rent for a particular space type, indicating the name and parameter values for the local-level effect; this is not included in the previous version of the SD Module:
  - LLEffectName; text characters of name of local-level effect;

- LLEffectCode; integer value of index for local level effect; this is g in equations 70, 71 and 72 in the Theoretical Formulation document;
- LLEffectType; text characters of indication of type of local-level effect, 'inc' indicates the rent for the space type increases with increasing distance from the effect, 'dec' indicates the rent for the space decreases with increasing distance from the effect;
- MaximumDistanceEffect; real value of maximum distance for local-level effect; this is $DistMax_g$ in equations 71 and 72 in the Theoretical Formulation document;
- LLEffectPowerFactor; real value of power parameter for local-level effect; this is $\psi_{g,h}$ in equations 71 and 72 in the Theoretical Formulation document;

- ZoningSchemesI; this specifies the nature of the zoning schemes that can be assigned to parcels in order to represent zoning rules, each zoning scheme includes a set of permitted space types that can be developed along with the range of permitted densities for these types and the relevant development fees and costs; each row concerns a specific permitted space type included in a specific zoning scheme, indicating the names and values for the combination; the full set of permitted space types for a given zoning scheme is defined by the set of rows with the same value for the zoning scheme index; zoning schemes are used in the previous version of the SD Module – with the schemes defined in the direct inputs to the SD Module and the specific scheme assigned to each parcel in the parcel database; in the current version of the SD Module the permitted space types and quantities and relevant development fees are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the zoning schemes specified here are not used;
    - ZoningScheme; text characters for name of zoning scheme;
    - ZoningSchemeCode; integer value of index for zoning scheme; all rows

with the same value for this index together define the full set of permitted space types for the zoning scheme;

- AllowedDevelopmentType; text characters for name of permitted space type; this must match exactly the commodity name used for this space type in its representation as a commodity in the AA Module;
- DevTypeCode; integer value of index for permitted space type; this must match the value of GridCodeInt for the space type as defined in DevelopmentTypesI;
- Units; text characters of units used for expressing quantities of permitted space type; this must match exactly the units designation for this space type as defined in DevelopmentTypesI;
- MaxIntensity; real value of maximum quantity of space type permitted on the parcel, in units of space type per unit of land; note that a dimensionless FAR value will have to be converted into the relevant space units per land unit; this is $MaxArea_h$ in equations 52 and 53 in the Theoretical Formulation document;
- MinIntensity; real value of minimum quantity of space type permitted on the parcel, in units of space type per unit of land; note that a dimensionless FAR value will have to be converted into the relevant space units per land unit; this is $MinArea_h$ in equations 52 and 53 in the Theoretical Formulation document;
- SpaceFee; real value of fee rate per unit space used to calculate one time money fees charged for development of space type, in units of money per unit of space type; this value is used to calculate $TrCostsS_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the development fees inputs to $TrCostsS_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- LandFee; real value of fee rate per unit land used to calculate one time

money fees charged for development of space type; in units of money per unit of land on the parcel; this value is used to calculate $TrCostsL_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the development fees inputs to $TrCostsL_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;

- QuantityFee; real value of cost rate per unit of services used to calculate one time money costs for providing services required for development of space type; in units of money per unit of services required for the parcel; the quantity of required services for the parcel is determined using the ServiceRequirement value specified in DevelopmentTypesI; this value is used to calculate $TrCostsL_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the service provision costs inputs to $TrCostsS_{v,h}$ and $TrCostsL_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;

- ServiceCostCodeI; this specifies the nature of the service cost categories that can be assigned to parcels in order to represent the costs for developing required services as part of development actions, each row concerns a specific service cost category for a specific space type, indicating the names and the value for the cost rate for providing services per unit of space type for the combination; service cost categories are used in the previous version of the SD Module – with the service cost categories defined in the direct inputs to the SD Module and the specific category assigned to each parcel in the parcel database – they provide a method for specifying costs associated with development options at the parcel-level that is an alternative to the one based on the specification of a QuantityFee rate for services (in ZoningSchemes) and a ServiceRequirement rate for the space type (in DevelopmentTypesI); in the current version of the SD Module the permitted space types and quantities and relevant development fees are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the zoning schemes specified here are not used:

- ServiceCostCodeDescription; text characters for name of service cost category;
- ServiceCostCode; integer value of index for service cost code;
- DevelopmentType; text characters of name of space type; this must match exactly the DevelopmentTypeName used for this space type in DevelopmentTypesI;
- Units; text characters of units used for expressing quantities of space type; this must match exactly the units designation for this space type as defined in DevelopmentTypesI;
- ServicingCost; real value of cost rate per unit of space for services used to calculate one time money costs for providing services required for development of space type; in units of money per unit of space; this value is used to calculate $TrCostsS_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the service provision costs inputs to $TrCostsS_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;

- TransitionConstantsI; a two-dimensional matrix where the rows concern existing space types and the columns concern updated space types; each cell in the matrix contains the value of the transition constant for the existing space type for the row of the cell transitioning into the updated space type for the column of the cell; these are the values of $TrConst_{v,h}$ in equation 52 of the Theoretical Formulation document, v is the row value and h is the column value for the cell.

These input files are included in Flow F in Figure 3.

4.5 Parcel Attribute Database

The PA database includes for each parcel (specifically, each pseudo-parcel) considered by the SD Module:

- fixed attributes that do not change from one year to the next in a model run;
    - unique ID number
    - land use zone (LUZ) containing the parcel, this is z in the Theoretical Formulation document;
    - transport analysis zone (TAZ) containing the parcel, this is i in the Theoretical Formulation document;
    - land area; this is LotSize in equations 52 and 53 in the Theoretical Formulation document;
    - shape of parcel for graphical presentation;
- varying attributes that can change from one year to the next in a model run;
    - existing space type; this is v in the Theoretical Formulation document; for years other than the base year this is the updated space type h* in the previous year, for the base year this is established using the base year PA-GUI as described below;
    - existing space quantity; this is $EArea_v$ in equation 55 in the Theoretical Formulation document; for years other than the base year this is the value for $UArea_{h^*}$ in the previous year, for the base year this is established using the base year PA-GUI as described below;
    - existing space age; this is Age_v in the Theoretical Formulation document; for years other than the base year this is calculated by adding 1 to the value for the relevant Age_{h^*} in the previous year; for the base year this is established using the base year PA-GUI as described below;
    - density in local area around parcel; this is Den in equation 70 in the Theoretical Formulation document;
    - distance from each local-level effect considered in model; this is $Dist_g$ in

equations 71 and 72 in the Theoretical Formulation document;

- land zoning policy and transformation cost attributes that can change from one year to the next in a model run;
  - indication whether developer event E0 is permitted;
  - indication whether developer event Er is permitted;
  - indication whether developer event Ed is permitted;
  - set of new space types permitted (with developer event En); this is H(n) in the Theoretical Formulation document;
  - minimum area for each permitted new space type, in units of space type; this is $MinArea_h$ in equations 52 and 53 in the Theoretical Formulation document;
  - maximum area for each permitted new space type, in units of space type; this is $MaxArea_h$ in equations 52 and 53 in the Theoretical Formulation document;
  - principal unit costs for transitioning to space type h per unit of space for each available updated space type h, in units of money per unit of space type; this is $InitCostsS_h$ in equation 74 in the Theoretical Formulation document;
  - ongoing unit costs for transitioning to space type h per unit of space per year for each available updated space type h; this is $OnCostsS_h$ in equation 74 in the Theoretical Formulation document;
  - principal unit costs for transitioning to space type h per unit of land for each available updated space type h, taking account of the existing space type v and servicing; this is $InitCostsL_{h,v}$ in equation 75 in the Theoretical Formulation document;
  - ongoing unit costs for transitioning to space type h per unit of land per year for each available updated space type h; this is $OnCostsL_h$ in equation 75 in the Theoretical Formulation document;

4.6 GUIs

Graphical user interfaces (GUIs) with the GIS parcel attribute (PA) database are included to provide both (a) map-based systems for specifying inputs to the PA database for use in model runs and (b) map-based presentations of results from model runs.

There are two GUIs for specifying inputs to the PA database. One GUI concerns the development of the base year PA database. The other GUI concerns the attributes that vary from year to year representing land zoning policy and changing development costs. Each of these GUIs is described below.

Base Year Parcel Attribute GUI

The base year parcel attribute GUI, also called the 'base year PA-GUI', is used to establish the values for the base year for the following attributes in the PA database:

- fixed attributes that do not change in a model run;
  - unique ID number
  - land use zone (LUZ) containing the parcel, this is z in the Theoretical Formulation document;
  - transport analysis zone (TAZ) containing the parcel, this is i in the Theoretical Formulation document;
  - land area; this is LotSize in equations 52 and 53 in the Theoretical Formulation document;
  - shape of parcel for graphical presentation;
- varying attributes that can change from one year to the next in a model run, that are values updated by the SD Module directly ;
  - initial existing space type for the base year; this is v in the Theoretical Formulation document;

- initial existing space quantity for the base year; this is $EArea_v$ in equation 55 in the Theoretical Formulation document;
- initial existing space age for the base year; this is Age_v in the Theoretical Formulation document;

These parcel attributes are included in Flow A in Figure 3.

The base year PA-GUI also establishes the pseudo-parcels, and thus requires ParcelMax as an input.

In general, the base year PA-GUI is run only in model development, to establish the base year version of the database for the model. At some point in model development, the base year version of the database is fixed and used as a common 'starting point' in every run of the model in application.

The functionality of the base year PA-GUI is as follows: ??update??

- system of layers

All Year Parcel Attribute GUI

The all year parcel attribute GUI, also called the 'all year PA-GUI', is used to establish the values for each year, from the base year to the final year of the simulation, for the following land zoning policy and transformation cost attributes in the PA database that can change from one year to the next in a model run:

- indication whether developer event E0 is permitted;
- indication whether developer event Er is permitted;
- indication whether developer event Ed is permitted;
- set of new space types permitted (with developer event En); this is H(n) in

the Theoretical Formulation document;

- minimum area for each permitted new space type, in units of space type; this is $MinArea_h$ in equations 52 and 53 in the Theoretical Formulation document;
- maximum area for each permitted new space type, in units of space type; this is $MaxArea_h$ in equations 52 and 53 in the Theoretical Formulation document;
- principal unit costs for transitioning to space type h per unit of space for each available updated space type h; $InitCostsS_h$; in equation 74 in the Theoretical Formulation document;
- ongoing unit costs for transitioning to space type h per unit of space per year for each available updated space type h; $OnCostsS_h$; in equation 74 in the Theoretical Formulation document;
- ongoing unit costs for transitioning to space type h per unit of land per year for each available updated space type h; $OnCostsL_h$ in equation 75 in the Theoretical Formulation document

These parcel attributes are included in Flow B in Figure 3.

In general, the all year PA-GUI is run with every model run in application, to establish the inputs to the macros, the database and the SD Module for each year consistent with the model scenario being considered in the model run.

The functionality of the all year PA-GUI is as follows: ??update??

- system of layers

4.7 Parcel Attribute GIS Macros

Marcos are included to update certain parcel attributes in the parcel database for each year using functions available with the GIS, called ‘PA-GIS Macros’. The parcel attributes updated with these macros are the ones that change in response to the existing conditions on parcels for each year, including:

- principal unit costs for transitioning to space type h per unit of land for each available updated space type h, taking account of the existing space type v and servicing; this is $InitCostsL_{h,v}$ in equation 75 in the Theoretical Formulation document;
- density in local area around parcel; this is Den in equation 70 in the Theoretical Formulation document;
- distance from each local-level effect considered in model; this is $Dist_g$ in equations 71 and 72 in the Theoretical Formulation document.

These parcel attributes are included in Flow D in Figure 3. Components of information required by the PA-GIS Macros to determine $InitCostsL_{h,v}$ for each parcel for each year are included in Flow C in Figure 3.

The functionality of the PA-GIS Macros is as follows: ??update??

5. SD-A Module Architecture

5.1 SD-A Module Software Code

The SD Module is written in Java.

The SD Module runs in close association with a GIS database containing the attributes for the full set of parcels included in the model system, called the 'parcel attribute database' or 'PA database'. One of several different commercially available GIS software packages can be used to implement and to access and manipulate the PA database. Certain macros and GUIs that work with this GIS system are also included to perform routines required as part of the interaction with the SD Module.

GIS software packages that have been used in previous installations, for which some macros and GUIs may already have been established, are:
ArcGIS 9 in combination with MySQL;
?? update ??

5.2 External Information Flows For SD-A Module

Figure 4 shows the information flows in and out of the SD Module and the associated GIS and GUIs in the step from a given year t to year t+1 in the full set-up. It indicates the origins and destinations of these information flows and also upper-case letter codes that are used to identify specific files and variables included in these flows. Table 3 provides a summary of these flows and their components using the letter codes in Figure 4.

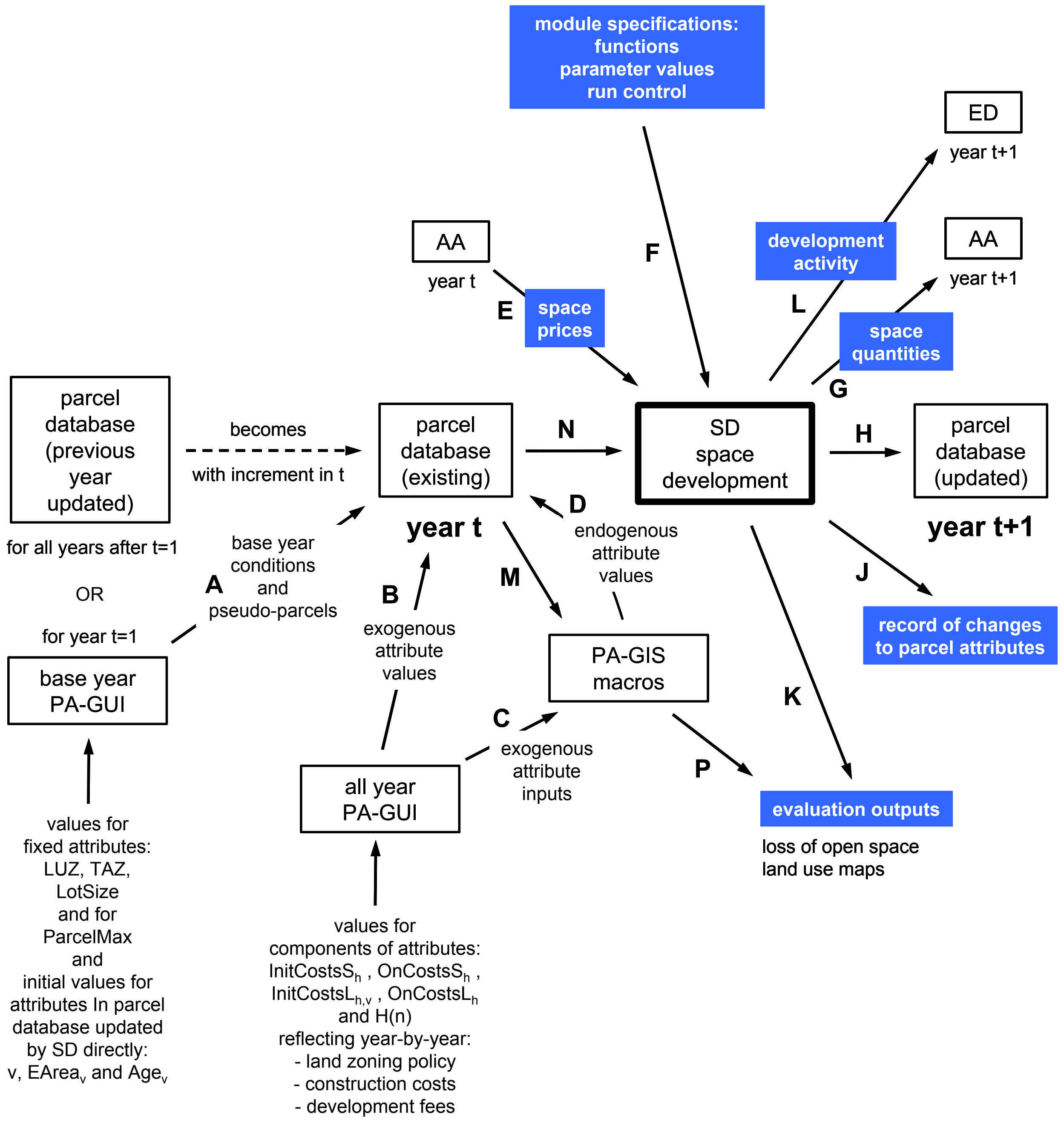

module specifications: functions parameter values run control
ED
year t+1
AA
year t
F
development activity
AA
year t+1
E
space prices
L
space quantities
G
parcel database (previous year updated)
becomes
with increment in t
parcel database (existing)
N
SD space development
H
parcel database (updated)
D
year t
year t+1
for all years after t=1
base year conditions and pseudo-parcels
endogenous attribute values
A
M
J
OR
B
record of changes to parcel attributes
for year t=1
exogenous attribute values
PA-GIS macros
base year PA-GUI
K
C
exogenous attribute inputs
P
all year PA-GUI
evaluation outputs
loss of open space
land use maps
values for fixed attributes: LUZ, TAZ, LotSize and for ParcelMax and initial values for attributes In parcel database updated by SD directly: v, EArea$_v$ and Age$_v$
values for components of attributes: InitCostsS$_h$, OnCostsS$_h$, InitCostsL$_{h,v}$, OnCostsL$_h$ and H(n) reflecting year-by-year:
- land zoning policy
- construction costs
- development fees

Figure 4: Information flows in and out of the SD Module and associated GUI and GIS Macro components in the step from a year t to a year t+1 put in the ‘full’ set-up where the SD Module is run in each year-to-year step in combination with the other system modules. Table 2 lists the components of these information flows using the letter codes (A through L) shown.

Table 3: Component files and variables in SD Module information flows in Figure 3

Flow in Figure 3	File	Variable
F	DevelopmentTypesI	
F	LocalLevelEffectsI	
F	ZoningSchemesI	
F	ServiceCostCodesI	
F	TransitionConstantsI	
A		unique ID for each parcel
A		land use zone (LUZ) for each parcel
A		transport analysis zone (TAZ) for each parcel
A		land area, LotSize, for each parcel
A		initial existing space type, v, for each parcel for base year
A		initial existing space quantity, $EArea_v$, for each parcel for base year
A		initial existing space age, Age_v, for each parcel for base year
H		existing space type, v, for each parcel for each year after base year
H		existing space quantity, $EArea_v$, for each parcel for each year after base year
H		existing space age, Age_v, for each parcel for each year after base year

B	indication whether developer event E0 is permitted for each parcel for each year
B	indication whether developer event Er is permitted for each parcel for each year
B	indication whether developer event Ed is permitted for each parcel for each year
B	set of new space types permitted (with developer event En), H(n), for each parcel for each year
B	minimum area for each permitted new space type, $MinArea_h$, for each parcel for each year
B	maximum area for each permitted new space type, $MaxArea_h$, for each parcel for each year
B	principal unit costs for transitioning to each available updated space type per unit of space, $InitCostsS_h$, for each parcel for each year
B	ongoing unit costs for transitioning to each available updated space type per unit of space, $OnCostsS_h$, for each parcel for each year
B	ongoing unit costs for transitioning to each available updated space type per unit of land, $OnCostsL_h$, for each parcel for each year
D	principal unit costs for transitioning to each available updated space type per unit of land, $InitCostsL_{h,v}$, for each parcel for each year
D	density in local area, Den, for each parcel for each year

D	distance from each local-level effect, $Dist_g$ for each parcel for each year

5.3 SD-A Module Run Control File

-- ld.properties ??DDS check and correct and expand as required – so it is like the treatment for aa.properties (see section 4.6 above) ??

When the SD Module is run, aspects of the algorithms it is to perform and the locations of certain inputs and outputs it is to use are specified in the form of ‘run controls’. These run controls are included in the ‘ld.properties’ file for the run, which is located in the sub-directory where the run is initiated. The values for certain model parameters are also set in the ‘ld.properties’ file.

A set-up for running the full model system with multiple runs of the SD Module for multiple years includes multiple ld.properties files, one in each of the directories where the SD Module is run.

The run controls are specified in the ld.properties file by setting program variables equal to specific values, with one such ‘equation’ setting per line and the lines in any order in the file. A comment line is indicated with ‘#’ in the first space of the line. The variables and settings are as follows:

LandJDBCDriver ??JEA: check this??

This does ???; an example setting is:

```
LandJDBCDriver=org.sqlite.JDBC
```

InputJDBCDriver=sun.jdbc.odbc.JdbcOdbcDriver

```
InputJDBCDriver=sun.jdbc.odbc.JdbcOdbcDriver
```

LandDatabase

```
LandDatabase=jdbc:sqlite:C:/BaltimorePECAS2005/SQLiteTest/sqliteparcels.db
```

InputDatabase=jdbc:odbc:BaltimorePECAS

```
InputDatabase=jdbc:odbc:BaltimorePECAS
```

ExcelInputDatabase

```
ExcelInputDatabase=true
```

ExcelLandDatabase

```
ExcelLandDatabase=false
```

LandTable

```
LandTable=parcels
```

UseSQLInputs

```
UseSQLInputs=true
```

UseSQLParcels

```
UseSQLParcels=true
```

DevelopmentDispersionParameter

DevelopmentDispersionParameter=.0000005

DevelopmentAlternativesDispersionParameter

DevelopmentAlternativesDispersionParameter=0.0000005

IntensityDispersionParameter

This sets the value for the utility function dispersion parameter for the allocation of updated space quantities, λ_{q,h^*} in equations 67 through 69 in the Theoretical Formulation document; an example setting is:

IntensityDispersionParameter=.000001

Note that the same single value is specified for all space types h.

LUZColumnName

This sets the label used in the parcel database for the land use zone (LUZ) attribute for the parcel (indicating the LUZ in which the parcel is located) in each parcel record; an example setting is:

LUZColumnName=ID3

where the label is ’ID3’

TAZColumnName

This sets the label used in the parcel database for the transport analysis zone (TAZ) attribute for the parcel (indicating the TAZ in which the parcel is located) in each parcel record; an example setting is:

```
TAZColumnName=ID2
```

where the label is ’ID2’

CoverageColumnName

This sets the label used in the parcel database for the space type attribute for the parcel (indicating the type of space developed on the parcel) in each parcel record; an example setting is:

```
CoverageColumnName=DevelopmentTypeCode
```

where the label is ‘DevelopmentTypeCode’

QuantityColumnName

This sets the label used in the parcel database for the space quantity attribute for the parcel (indicating the quantity of space developed on the parcel) in each parcel record; an example setting is:

```
QuantityColumnName = AmountOfDevelopment
```

where the label is ‘AmountOfDevelopment’

ZoningColumnName

```
ZoningColumnName=ZoningSchemeCode
```

This sets the label used in the parcel database for the zoning scheme attribute code used to identify the specific set of zoning rules for the parcel in each parcel record; an example setting is:

ZoningColumnName = ZoningSchemeCode

where the label is ‘ZoningSchemeCode’

SizeColumnName

This sets the label used in the parcel database for the quantity of land attribute for the parcel in each parcel record; an example setting is:

SizeColumnName = AmountOfLand

where the label is ‘AmountOfLand’

YearBuiltColumnName

This sets the label used in the parcel database for the year built attribute for the space on the parcel in each parcel record; an example setting is:

YearBuiltColumnName = YearBuilt

where the label is ‘YearBuilt’

AAResultsDirectory

AAResultsDirectory = C:/BaltimorePECAS2005/Baltimore PECAS InAndOut/

LogFilePath

LogFilePath = C:/BaltimorePECAS2005/Baltimore PECAS InAndOut/

IntegerCodeForCoverage

```
IntegerCodeForCoverage = true
```

MaxParcelSize

```
MaxParcelSize = 1
```

InputParcelFile

```
InputParcelFile = C:\\BaltimorePECAS2005\\SQLiteTest\\sqlitebulkload.sql
```

OutputParcelFile

This sets the directory path and filename for the file where the SD Module outputs are written; forward slashes are used (in place of the standard backward slashes); an example setting is:

```
OutputParcelFile = C:/Pecas/BaseRun/BaseOutputs/SDResults.csv
```

where the directory path is " C:\Pecas\BaseRun\BaseOutputs\SDResults.csv "

5.4 SD-A Module Input Files and Variables

The structure and contents of the input files for the SD Module are indicated below.

- DevelopmentTypesI Each row concerns a specific space type, indicating the name and parameter values for the space type:
  - DevelopmentTypeName; text characters of name of space type; this must match exactly the commodity name used for this space type in its representation as a commodity in the AA Module;

- GridCodeInt; integer value of index for space type, used for both v and h indices for existing and updated space types;
- Units; text characters of units used for expressing quantities of space type; this must match exactly the units designation for the commodity corresponding to this space type as specified in the CommoditiesI input file for the AA Module;
- ConstructionCost; real value of unit cost for construction of new space type, for the En development event, in units of money per unit of space type; this value is used to calculate the $TrCostsS_{v,h}$ and $TrCostsL_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the construction cost inputs to $TrCostsS_{v,h}$ and $TrCostsL_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- ReconstructCost; real value of unit cost for construction of altered or renovated space type, for the Er development event, in units of money per unit of space type; this value is used to calculate the $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the reconstruction cost inputs to $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- MaintenanceCost; real value of base unit costs for maintaining space type (when it has an age of 1 year); expressed as an amount per unit of space per year; this is $MtBase_h$ in equation 76 in the Theoretical Formulation document;
- AddASC; real value of transition constant for ‘alter or renovate’ development event Er for the space type; this is $TrConst_{v,h=vEr}$ in the Theoretical Formulation document;
- SameASC; real value of transition constant for ‘no change’ development

event E0 for the space type; this is $TrConst_{v,h=vE0}$ in equation 55 in the Theoretical Formulation document;

- AgeRentDiscount; real value of proportional decrease in rent for space type with each additional year of age; expressed as a proportion without units; this is $AgeFac_h$ in equation 70 in the Theoretical Formulation document;,
- DensityRentDiscount; real value of proportional decrease in rent for space type with each additional increase in density in the area around the parcel; expressed as a proportion without units; this is $DenFac_h$ in equation 70 in the Theoretical Formulation document;,
- RegPriceEffect; ??update??
- RegPriceReference; ??update??
- LandRent; ??update??
- TypicalFAR; ??update??
- AgeMaintenanceCost; real value of proportional increase in maintenance costs for space type with each additional year of age; expressed as a proportion without units; this is $MtFac_h$ in equation 76 in the Theoretical Formulation document;
- DemolitionCost; real value of unit cost for demolition of existing space type, in units of money per unit of space type; this value is used to calculate the $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the demolition cost inputs to $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- ServiceRequirement; real value of rate at which services are required for space type, in units of money per unit of space type; this value is used to calculate the $TrCostsS_{v,h=vEr}$ and $TrCostsL_{v,h=vEr}$ within the SD Module in the previous version of the SD Module; in the current version of the SD

Module the servicing cost inputs to TrCostsS$_{v,h=vEr}$ and TrCostsL$_{v,h=vEr}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;

- VacantType; ??update??

- LocalLevelEffectsI; each row concerns a specific local-level effect on the rent for a particular space type, indicating the name and parameter values for the local-level effect; this is not included in the previous version of the SD Module:
  - LLEffectName; text characters of name of local-level effect;
  - LLEffectCode; integer value of index for local level effect; this is g in equations 70, 71 and 72 in the Theoretical Formulation document;
  - LLEffectType; text characters of indication of type of local-level effect, 'inc' indicates the rent for the space type increases with increasing distance from the effect, 'dec' indicates the rent for the space decreases with increasing distance from the effect;
  - MaximumDistanceEffect; real value of maximum distance for local-level effect; this is DistMax$_g$ in equations 71 and 72 in the Theoretical Formulation document;
  - LLEffectPowerFactor; real value of power parameter for local-level effect; this is $\psi_{g,h}$ in equations 71 and 72 in the Theoretical Formulation document;

- ZoningSchemesI; this specifies the nature of the zoning schemes that can be assigned to parcels in order to represent zoning rules, each zoning scheme includes a set of permitted space types that can be developed along with the range of permitted densities for these types and the relevant development fees and costs; each row concerns a specific permitted space type included in a specific zoning scheme, indicating the names and values for the combination; the full set of permitted space types for a given zoning scheme is defined by the set of rows with the same value for the zoning scheme index; zoning schemes are used in the previous version of the SD Module – with the schemes defined in the direct inputs to the SD Module and the specific scheme assigned to each parcel in the parcel database; in the current version of the SD Module the permitted space types and quantities and relevant development fees are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the zoning schemes specified here are not used;

- ZoningScheme; text characters for name of zoning scheme;
- ZoningSchemeCode; integer value of index for zoning scheme; all rows with the same value for this index together define the full set of permitted space types for the zoning scheme;
- AllowedDevelopmentType; text characters for name of permitted space type; this must match exactly the commodity name used for this space type in its representation as a commodity in the AA Module;
- DevTypeCode; integer value of index for permitted space type; this must match the value of GridCodeInt for the space type as defined in DevelopmentTypesI;
- Units; text characters of units used for expressing quantities of permitted space type; this must match exactly the units designation for this space type as defined in DevelopmentTypesI;
- MaxIntensity; real value of maximum quantity of space type permitted on the parcel, in units of space type per unit of land; note that a dimensionless FAR value will have to be converted into the relevant space units per land unit; this is $MaxArea_h$ in equations 52 and 53 in the Theoretical Formulation document;
- MinIntensity; real value of minimum quantity of space type permitted on the parcel, in units of space type per unit of land; note that a dimensionless FAR value will have to be converted into the relevant space units per land unit; this is $MinArea_h$ in equations 52 and 53 in the Theoretical Formulation document;

- SpaceFee; real value of fee rate per unit space used to calculate one time money fees charged for development of space type, in units of money per unit of space type; this value is used to calculate TrCostsS$_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the development fees inputs to TrCostsS$_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- LandFee; real value of fee rate per unit land used to calculate one time money fees charged for development of space type; in units of money per unit of land on the parcel; this value is used to calculate TrCostsL$_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the development fees inputs to TrCostsL$_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;
- QuantityFee; real value of cost rate per unit of services used to calculate one time money costs for providing services required for development of space type; in units of money per unit of services required for the parcel; the quantity of required services for the parcel is determined using the ServiceRequirement value specified in DevelopmentTypesI; this value is used to calculate TrCostsL$_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the service provision costs inputs to TrCostsS$_{v,h}$ and TrCostsL$_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;

- ServiceCostCodeI; this specifies the nature of the service cost categories that can be assigned to parcels in order to represent the costs for developing required services as part of development actions, each row concerns a specific service cost category for a specific space type, indicating the names and the value for the cost rate for providing services per unit of space type for the combination; service cost categories are used in the previous version of the SD Module – with the service cost categories defined in the direct inputs to the SD Module and the specific category assigned to each parcel in the parcel database – they provide a method for specifying costs associated with development options at the parcel-level that is an alternative to the one based on the specification of a QuantityFee rate for services (in ZoningSchemes) and a ServiceRequirement rate for the space type (in DevelopmentTypesI); in the current version of the SD Module the permitted space types and quantities and relevant development fees are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the zoning schemes specified here are not used:

- ServiceCostCodeDescription; text characters for name of service cost category;
- ServiceCostCode; integer value of index for service cost code;
- DevelopmentType; text characters of name of space type; this must match exactly the DevelopmentTypeName used for this space type in DevelopmentTypesI;
- Units; text characters of units used for expressing quantities of space type; this must match exactly the units designation for this space type as defined in DevelopmentTypesI;
- ServicingCost; real value of cost rate per unit of space for services used to calculate one time money costs for providing services required for development of space type; in units of money per unit of space; this value is used to calculate $TrCostsS_{v,h}$ within the SD Module in the previous version of the SD Module; in the current version of the SD Module the service provision costs inputs to $TrCostsS_{v,h}$ are provided at the parcel level via the all year PA-GIS and the PA-GIS macros, so the value specified here is not used;

- TransitionConstantsI; a two-dimensional matrix where the rows concern existing space types and the columns concern updated space types; each cell in the matrix contains the value of the transition constant for the existing space type for the row of the cell transitioning into the updated space type for the column of the cell; these are the values of $TrConst_{v,h}$ in equation 52 of the Theoretical Formulation document, v is the row value and h is the column value for the cell.

These input files are included in Flow F in Figure 3.

6. Set-up and Installation

6.1 Two Forms of Set-up

There are two forms of set-up for PECAS.

One form of set-up is used when running the full model system through time. This occurs when using a completed model in policy analysis and also when in stage three (the final stage) of calibration. It is referred to as the ‘full’ set-up.

The other form of set-up is used in stage two of calibration, when the different system modules are being run and worked on separately.

In stage two of calibration of the AA Module, the AA Module is run on its own with inputs that are developed externally, not taken directly from the other modules. Some of the inputs for the AA Module in stage two are specified in Excel spreadsheets and the system set-up is designed to run the AA Module on its own in just one of the year directories, typically the base year for the model.

The same is the case in stage two of calibration of the SD Module: some of the inputs are specified in Excel spreadsheets and the system set-up is designed to run the SD Module on its own in just one of the year directories. These are referred to as the ‘separate’ set-ups for the AA Module and the SD Module.

The SD Module runs in close association with a database containing the attributes for the full set of parcels included in the model system, called the ‘parcel attribute database’ or ‘PA database’ as described in section 4.1 and several different commercially available database software packages can be used to implement and then to access and manipulate the PA database. The SQLite software package is used here in this illustration of setup and installation.

Both the ‘full’ set-up and the ‘separate’ set-up are described in this document.

6.2 Directory Structure

The components of the directory structure and the types of files they contain for running the AA and SD Modules, both on their own and as part of the full model system, are described below. These components of the directory structure need to be created as part of the installation of the PECAS software.

In these descriptions, the italicized text characters *‘Project’* are used to indicate a text description of the specific model project or area being considered.

Program Files

The program files containing the machine code for running the AA and SD Modules are located in the directory ‘C:\Program Files\PECAS*Project*’. For example, for a PECAS model of San Diego, the program files are located in the directory ‘C:\Program Files\PECASSanDiego’.

Run Files

The run files are the files other than the program files needed to run the AA and SD Modules, including the input files, databases and run control files. These run files are located in sub-directories under the directory ‘C:\PECAS*Project*\‘. The general structure of these sub-directories is depicted in Figure 5.

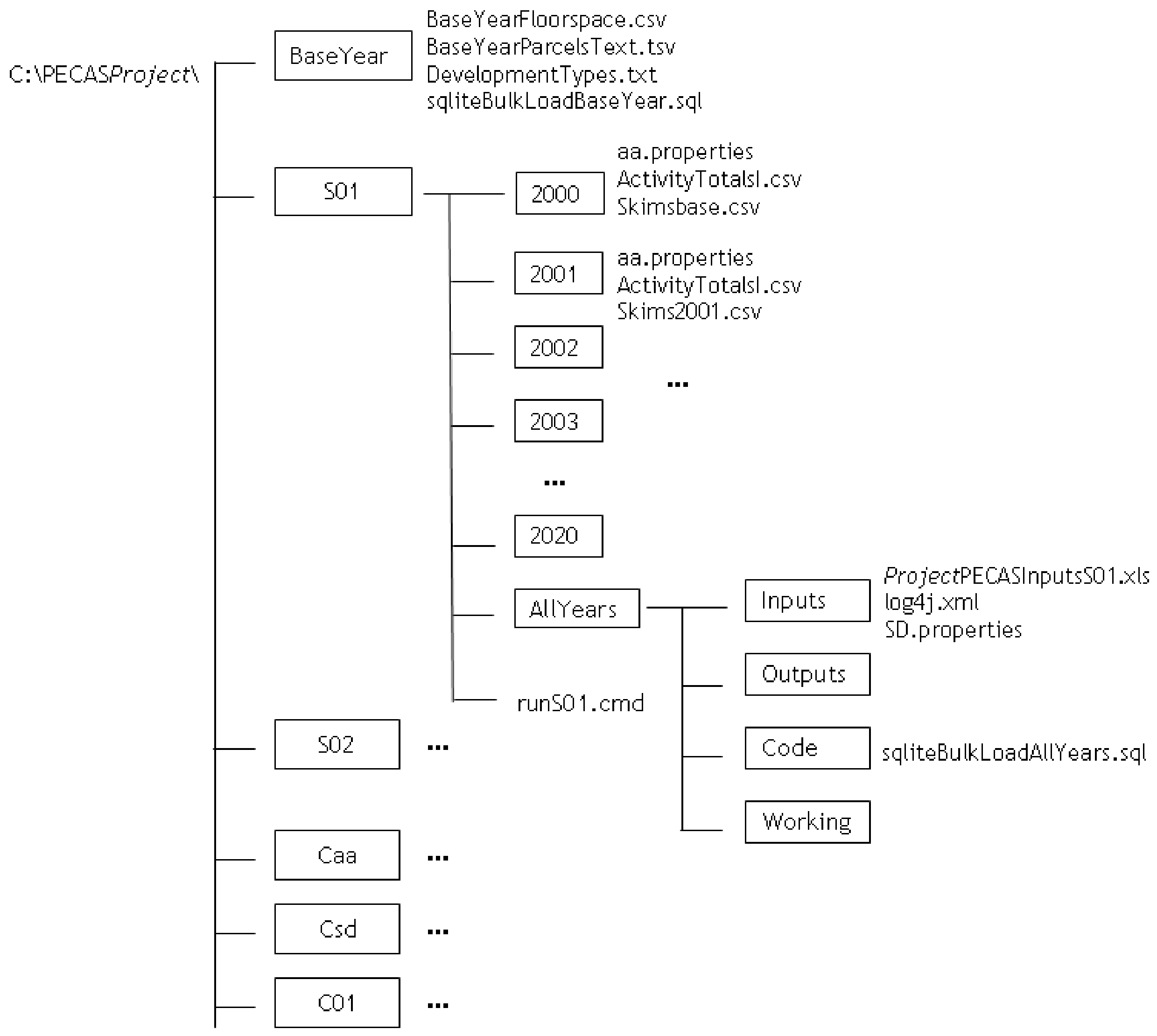


Figure 5: Portion of directory structure for running the AA and SD Modules, both on their own and as part of the full model system. The portion shown is for a scenario 'S01' for the years 2000 through 2020 inclusive. The relevant run files are located in the sub-directories within this structure.

Scenarios

A run of the full model system through time (using the ‘full’ set-up) with a specific set of inputs reflecting both policy and non-policy conditions is called a ‘scenario’. Each such scenario for a project is assigned a label ‘Snn’, where ‘nn’ is a unique 2-digit number. The run files specific to a given scenario Snn are included in a series of sub-directories under the directory ‘C:\PECAS*Project*\Snn’.

The files for a specific year in the scenario are included in a sub-directory with the year as its name within the sub-directory for the scenario.

The inputs and outputs common to all years in the scenario, and the program code used to run the scenario across the years, are included in specific sub-directories under an ‘AllYears’ sub-directory for the scenario.

The parcel database used by the SD Module is included in a specific ‘working’ sub-directory within the ‘AllYears’ sub-directory, along with any other working versions of files used in multiple years of a scenario run. If ‘snap-shot’ records of the parcel database for a specific year in the scenario are required, then a copy of the working version of the parcel database is retained for that year, with it copied into the corresponding sub-directory for that year for the scenario.

Base Year

The files concerning the base year starting point that is common to all scenarios are located in the sub-directory ‘C:\PECAS*Project*\BaseYear’. This includes, in general, the base year version of the parcel database or space quantities file, which is the fixed and common starting point for all scenarios, along with any files required to prepare a separate working version of this database file.

Calibration

The stage two calibration work is done under the directory ‘C:\PECAS*Project*\’, using the ‘separate’ set-up for AA and SD modules. Each such scenario considered in this calibration work is assigned a label ‘Cnn’, where ‘nn’ is ‘aa’ for the AA module calibration, ‘sd’ for SD module calibration and ‘01’ for the stage three (the final stage) of calibration of AA and SD module as a part a ‘full setup’ of the modal system. The specific directory for the stage two calibration runs of the AA Module is ‘C:\PECAS*Project*\Caa’; for the SD Module is ‘C:\PECAS*Project*\Csd’ and for stage three calibration in ‘C:\PECAS*Project*\C01’. Again, the scenario specific files for this calibration scenario labeled Cnn are located in a series of sub-directories under the directory ‘C:\PECAS*Project*\Cnn’, one such sub-directory for each year considered in the calibration and rest of the files common to all years are under sub directory ‘AllYears’ similar to a scenario setting.

The stage three calibration work is also done in sub-directories under the directory ‘C:\PECAS*Project*\’, using the ‘full’ set-up in one or more scenarios. Each such scenario considered in this calibration work is assigned a label ‘Cnn’, where ‘nn’ is a unique 2-digit number. Again, the scenario specific files for this scenario labeled Cnn are located in a series of sub-directories under the directory ‘C:\PECAS*Project*\Cnn’, one such sub-directory for each year considered in the calibration scenario. The inputs and outputs common to all years in the scenario, and the program code used to run the scenario across the years, are included in specific sub-directories under an ‘AllYears’ sub-directory for the scenario.

The first (and primary) scenario considered in this ‘full’ setup calibration work is labeled C01 (that is, with nn=01). Other scenarios considered in this work, perhaps as part of the examination of certain system elasticities over time, are labeled with nn values greater than 01.

6.3 Files

The files located in the sub-directories for running the AA Module and the SD Module both alone (in the 'separate' set-up) and as part of the full model system (in the 'full' set-up) are indicated below. These files need to be placed as indicated as part of the installation of the PECAS software and the implementation of a specific model using the software.

In these descriptions, the italicized text characters *'Project'* are used to indicate a text description of the specific model project or area being considered.

AA Module and SD Module Program Files

The program files for the AA and SD Modules are located in the sub-directory 'C:\Program Files\PECAS*Project*'. These program files include:

1: A set of files containing the Java computer code for the AA and SD Modules in particular:

*Project*pecas.jar
censusdata.jar
common-daf-V2.jar
pecas.jar
log4j-1.2.9.jar
or124.jar
mtj.jar
common-base.jar
sqlitejdbc-v033-native.jar

2: A set of files required for running the SQLite software that establishes the parcel database for the SD Module:

sqlite.exe

sqlite3.dll

sqlitejdbc.dll

These files are required for running the SQLite software for the parcel database.

Alternatively, the files sqlite.exe and sqlite3.dll can be placed in the directory 'C:\Windows\system32', (if the user has administrative privileges) in which case the Path Environment Variable specification indicated in section 6.4 below is not required.

AA Module 'Separate' Set-up Run Files

The scenario considered here is an AA Module calibration for year 2000.

The run files for the separate set-up of the AA Module are located in the sub-directory 'C:\PECAS*Project*\Caa'. These run files include:

1: The input file containing the specification inputs and parameter values for the AA Module for the stage two calibration work under the sub-directory 'C\PECAS*Project*\Caa\AllYears\Inputs':

*Project*PECASInputsSepAA.xls

The form and content of this file are described in section 4.7 in this document.

2: The input files containing the input data for the AA Module for the stage two calibration work under the sub –directory ‘C\PECAS*Project*\Caa\2000’:

ActivityTotalsI.csv

FloorspaceI.csv

Skims.csv

The data in these files represent the conditions for the inputs to the AA Module consistent with the calibration targets being used in the stage two calibration work. This includes the model-wide activity totals and quantities of space by type in each zone for the current year and the transport utilities among zones for the previous year. The calibration work is thus the task of adjusting the model parameter values and functional forms in order to improve the match between the targets and the corresponding model outputs arising with the inputs included in these files listed here.

3: The run controls file for the AA Module under the sub-directory ‘C\PECAS*Project*\Caa\2000’:

aa.properties

This file contains specifications regarding aspects of the algorithms the AA Module is to perform and the locations of certain inputs and outputs it is to use in this ‘separate’ set-up in particular. The values for certain model parameters are also set in this file. The full set of variables and possible settings for this file are described in section 4.6 in this document.

The specific settings required in this ‘separate’ set-up for the new version of AA module (Technology Options) are as follows:

output.data= C:/PECAS*Project*/Caa/2000

```
aa.base.data=C:/PECASProject/Caa/2000
aa.current.data= C:/PECASProject/Caa/2000
processor.class = com.pb.models.pecas.AASetupWithTechnologySubstitution
aa.useLogittechnologyChoice = true
aa.useLogitProduction =true
aa.calculateAveragePrices=true
aa.maxIterations = 300
aa.datasource = jdbc:odbc:ProjectPECASInputsSepAA
aa.jdbcDriver = sun.jdbc.odbc.JdbcOdbcDriver
aa.excelInputs = true
aa.useSQLInputs = true
aa.initialStepSize = .001
aa.minimumStepSize = .1
aa.maximumStepSize = 1.0
aa.ConFac= 0.001
aa.maxTotalClearance = 0.001
aa.maxSpecificClearance = 0.001
aa.localPriceStepSizeAdjustment = 0.5
Model.skimFormat = TableDataSet
aa.useFloorspaceZones = false
aa.writeUtilityComponents=false
skim.data= C:/PECASProject/Caa/2000
skim.filename=Skims.csv
aa.logFrequency = 50

constrained =false
constraint.iterations=10
constraint.smoothing=1.0
constraint.maxConstantChange=2.5
```

4: The batch files containing the commands for running the AA Module in a ‘separate‘ set-up in this sub-directory ‘C:\PECAS*Project*\Caa’ as intended:

RunAASep.cmd

RunAASepConstrained.cmd

With these batch files in place, the AA Module is run (in a ‘separate’ set-up in this sub-directory) by entering the relevant portion of text before the extension .cmd from a DOS command window. The AA Module can be run with or without the activity constraints routine operating as described in section 4.5 in this document. For a run of the AA Module without the activity constraints routine, the portion of text ‘RunAASep’ is entered. For a run of the AA Module with the activity constraints routine, the portion of text ‘RunAASepConstrained’ is entered (for an activity constrained routine aa.properties should be changed to be ‘constrained=true’.)

The specific contents of these batch files are listed below. These are listed below in several lines of text for clarity. In the batch files themselves the material on these several lines must be listed all on one line.

RunAASep.cmd has the contents:

java –Xmx400M –Dlog4j.configuration=log4j.xml –cp

“C:\Program Files\PECAS*Project**Project*pecas.jar”;

“C:\Program Files\PECAS*Project*\censusdata.jar”;

“C:\Program Files\PECAS*Project*\common-daf-V2.jar”;

“C:\Program Files\PECAS*Project*\pecas.jar”;

“C:\Program Files\PECAS*Project*\log4j-1.2.9.jar”;

“C:\Program Files\PECAS*Project*\or124.jar”;

“C:\Program Files\PECAS*Project*\mtj.jar”;

“C:\Program Files\PECAS*Project*\common-base.jar”;

com.hbaspecto.pecas.AAControl 2000 1

RunAASepConstrained.cmd has the contents:

java –Xmx400M –Dlog4j.configuration=log4j.xml –cp
"C:\Program Files\PECAS*Project**Project*pecas.jar";
"C:\Program Files\PECAS*Project*\censusdata.jar";
"C:\Program Files\PECAS*Project*\common-daf-V2.jar";
"C:\Program Files\PECAS*Project*\pecas.jar";
"C:\Program Files\PECAS*Project*\log4j-1.2.9.jar";
"C:\Program Files\PECAS*Project*\or124.jar";
"C:\Program Files\PECAS*Project*\mtj.jar";
"C:\Program Files\PECAS*Project*\common-base.jar";
com.hbspecto.pecas.AAConstrainedControl 2000 1

SD Module 'Separate' Set-up Run Files

The run files for the separate set-up of the SD Module considered here is for an SD Module calibration for year 2000. These run files include:

1: The input file containing the specification inputs and parameter values for the SD Module for the stage two calibration work under the sub-directory 'C\PECAS*Project*\Csd\AllYears\Inputs':

*Project*PECASInputsSepSD.xls

The form and content of this file are described in section 5.4 in this document.

2: The input files containing the input data for the SD Module for the stage two calibration work under the sub –directory 'C\PECAS*Project*\Caa\2000': This includes the prices for space by type in each land use zone for the existing year.

ExchangeResults.csv

3: The input files containing the base year parcel input data for the SD Module for the stage two calibration work under the sub –directory 'C\PECAS*Project*\BaseYear':

BaseYearParcelText.tsv

DevelopmentTypes.txt'

The data in these files represent the conditions for the inputs to the SD Module consistent with the calibration targets being used in the stage two calibration work. This includes the descriptions of the parcels for the existing year along with the development types considered. The calibration work is thus the task of adjusting the model parameter values and functional forms in order to improve the match between the targets and the corresponding model outputs arising with the inputs included in these files listed here.

4: SQLite database file to store the parcel data base to run the SD module is created on C:\PECAS*Project*\Csd\AllYears\Working through a series of DOS commands.

sqliteparcelsSepSD.db

The SQLite database is created as a setup and install step and is done on a DOS command prompt with the directory path set to C:\PECAS*Project*\Csd\AllYears\Working so that the file is created in the desired location. (C:\> CD C:\PECAS*Project*\Csd\AllYears\Working).

The specific script required to create the file on the dos command is as follows:

```
sqlite3 sqliteparcelsSepSD.db
CREATE TABLE parcels (ID1 CHAR(15), ID2 INT, AmountOfLand FLOAT,
DevelopmentTypeCode CHAR, AmountOfDevelopment FLOAT, YearBuilt
SMALLINT, ZoningSchemeCode SMALLINT, AmountOfService FLOAT);
create index id1 on parcels (id1);
create index id2 on parcels (id2);
create index id2DevType on parcels (id2,developmenttypecode);
create table DevelopmentTypes (GridCode CHAR, DevelopmentTypeName
TINYTEXT);
.mode tab
.import ‘C:/PECASProject/BaseYear/DevelopmentTypes.txt' DevelopmentTypes
.exit
```

5: SQLite script file to load in the text version of the database to the sqltiteparcelsSepSD.db database as a part of SD module run orchestration: The file is under the sub directory ‘C:/PECAS*Project*/BaseYear’

sqliteBulkLoadBaseYear.sql

The specific script is as follows:

```
.mode tab
delete from parcels;
vacuum;
.import ‘C:/PECASProject/BaseYear/BaseYearParcelDatabase.tsv’ parcels
.exit
```

6: A blank file under the subdirectory 'C:/PECASProject/Csd/AllYears/Inputs'. This is required in order to specify what occurs in the logging process and in terms of what is sent to the screen and what is sent to the log file.

log4j.xml

7: The run controls file for the SD Module under the subdirectory 'C:/PECAS*Project*/Csd/AllYears/Inputs':

sd.properties

This file contains specifications regarding aspects of the algorithms the SD Module is to perform and the locations of certain inputs and outputs it is to use in this 'separate' set-up in particular. The values for certain model parameters are also set in this file. The full set of variables and possible settings for this file are described in section 5.3 in this document.

The specific settings required in this 'separate' set-up are as follows:

LandJDBCDriver=org.sqlite.JDBC
InputJDBCDriver=sun.jdbc.odbc.JdbcOdbcDriver
LandDatabase=jdbc:sqlite:C:/PECAS*Project*/Csd/AllYears/Working/sqliteparcelsSepSD.
db InputDatabase=jdbc:odbc:*Project*PECASInputsSepSD
ExcelInputDatabase=true
ExcelLandDatabase=false
LandTable=parcels

#******************These are for operation ****************
UseSQLInputs=true

UseSQLParcels=true

DevelopmentDispersionParameter=.0000005

DevelopmentAlternativesDispersionParameter=0.0000005

IntensityDispersionParameter=.000001

LUZColumnName=ID3

TAZColumnName=ID2

CoverageColumnName=DevelopmentTypeCode

QuantityColumnName=AmountOfDevelopment

ZoningColumnName=ZoningSchemeCode

SizeColumnName=AmountOfLand

YearBuiltColumnName=YearBuilt

AAResultsDirectory= C:/PECAS*Project*/Csd

SDOutputsDirectory= C:/PECAS*Project*/Csd

LogFilePath= C:/PECAS*Project*/Csd

IntegerCodeForCoverage=true

MaxParcelSize=1

InputParcelFile=C:\\PECAS*Project*/\\BaseYear\\sqliteBulkLoadBaseYear.sql

OutputParcelFile= C:/PECAS*Project*/Csd/AllYears/Working/parcelstxtout.tsv

8: The batch file containing the commands for running the SD Module in a ‘separate‘ set-up in the sub-directory ‘C:\PECAS*Project*\Csd’:

RunSDSep.cmd

With this batch file in place, the SD Module is run (in a ‘separate’ set-up in this sub-directory) by entering the relevant portion of text before the extension .cmd from a DOS command window – specifically: ‘RunSDSep’ is entered.

The specific content of this batch file is listed below. This is done in several lines of text for clarity. In the batch file itself the material on these several lines must be listed all on one line.

RunSDSep.cmd has the contents:

```
java -Xmx400M -Dlog4j.configuration=log4j.xml –cp .;
"C:\Program Files\pecas\pecas.jar";
"C:\Program Files\pecas\ProjectPecas.jar";
"C:\Program Files\pecas\common-base.jar";
"C:\Program Files\pecas\log4j-1.2.9.jar";
"C:\Program Files\pecas\censusdata.jar";
"C:\Program Files\pecas\or124.jar";
"C:\Program Files\pecas\sqlitejdbc-v033-nested.jar”;
com.hbaspecto.Project.SDModel2 2000 1
```

The output files for the separate set-up of the SD Module are also specified to be placed (by the SD Module) in the sub-directory ‘C:\PECAS*Project*\Csd’. Note that the updated parcel database is written to the file ‘parcelstxtout.tsv in order to preserve the existing database stored in ‘BaseYearParcelText.tsv for use in subsequent runs of the SD Module in this separate set-up as part of the stage two model calibration work.

‘Full’ Set-up Run Files

The ‘full’ setup is used as part of stage three calibration and a scenario run on a completed model in policy analysis. The run files setup below is in reference to a scenario run of AA and SD module running through time from year 2000 to 2020. The run files are located in the sub-directory ‘C:\PECAS*Project*\S01’ as shown in Figure 5. These run files include:

1: The input file containing the specification inputs and parameter values for the AA and SD Modules under the sub-directory ‘C\PECAS*Project*\S01\AllYears\Inputs’:

*Project*PECASInputsS01.xls

The form and content of this file are described in section 4.7 and section 5.4 in this document.

2: The input files containing the input data for the AA Module for each year considered in the system (in this case from 2000 to 2020). The following sets of files are included in each year folders (2000 to 2020) under the sub –directory ‘C\PECAS*Project*\S01\: There for the files for year 2005 is under the directory ‘C\PECAS*Project*\S01\2005’ are:

ActivityTotalsI.csv
Skims2005.csv
aa.properties

This includes the model-wide activity totals in each zone (ActvityTotalsI.csv) for that particular year and the transport utilities among zones for the previous year (Skims2005.csv). Note that the skims file for year 2000 (base year in this case) is Skimsbase.csv

The AA Module run controls file for each year considered in the system (aa.properties) contains specifications regarding aspects of the algorithms the AA Module is to perform and the locations of certain inputs and outputs it is to use in this ‘ful’ set-up in particular. The values for certain model parameters are also set in this file. The full set of variables and possible settings for this file are described in section 4.6 in this document.

The specific settings required in this ‘full’ set-up for the new version of AA module (Technology Options) are as follows: Note that the highlighted parts needs to be changed when in a particular folder (if in 2010 folder the highlighted to be 2010 instead of 2005)

```
output.data= C:/PECASProject/CS01/2005
aa.base.data=C:/PECASProject/CS01/2005
aa.current.data= C:/PECASProject/CS01/2005
processor.class = com.pb.models.pecas.AASetupWithTechnologySubstitution
aa.useLogittechnologyChoice = true
aa.useLogitProduction =true
aa.calculateAveragePrices=true
aa.maxIterations = 300
aa.datasource = jdbc:odbc:ProjectPECASInputsS01
aa.jdbcDriver = sun.jdbc.odbc.JdbcOdbcDriver
aa.excelInputs = true
aa.useSQLInputs = true
aa.initialStepSize = .001
aa.minimumStepSize = .1
aa.maximumStepSize = 1.0
aa.ConFac= 0.001
aa.maxTotalClearance = 0.001
aa.maxSpecificClearance = 0.001
aa.localPriceStepSizeAdjustment = 0.5
Model.skimFormat = TableDataSet
aa.useFloorspaceZones = false
aa.writeUtilityComponents=false
skim.data= C:/PECASProject/S01/2005
skim.filename=Skims2005.csv
aa.logFrequency = 50

constrained =false
```

```
constraint.iterations=10
constraint.smoothing=1.0
constraint.maxConstantChange=2.5
```

Note that aa.properties is the file that is amended for interaction with the transportation model. In a case where transportation model is run in only 5 year steps rather than each year (Transportation model running in 2005 for the first time from the base year and transportation utilities in each zones recorded in Skims2005.csv) the following will remain unchanged in aa.properties file in folder 2001 to 2004

```
skim.data= C:/PECASProject/S01/2005
skim.filename=Skims2005.csv
```

3: The input files containing the base year parcel input data for the SD Module for the stage two calibration work under the sub –directory 'C\PECAS*Project*\BaseYear':

BaseYearParcelText.tsv
BaseYearFloorSpace.csv
DevelopmentTypes.txt'

The data in these files includes the descriptions of the parcels for the base year in two formats; 1) comma separated values (csv) and 2) Tab separate values (tsv) along with the development types considered.

4: SQLite database file to store the parcel data base to run the SD module is created on C:\PECAS*Project*\S01\AllYears\Working through a series of DOS commands.

sqliteparcels.db

The SQLite database is created as a setup and install step and is done on a DOS command prompt with the directory path set to C:\PECAS*Project*\S01\AllYears\Working so that the file is created in the desired location. (C:\> CD C:\PECAS*Project*\S01\AllYears\Working).

The specific script required to create the file on the dos command is as follows:

```
sqlite3 sqliteparcels.db
CREATE TABLE parcels (ID1 CHAR(15), ID2 INT, AmountOfLand FLOAT,
DevelopmentTypeCode CHAR, AmountOfDevelopment FLOAT, YearBuilt
SMALLINT, ZoningSchemeCode SMALLINT, AmountOfService FLOAT);
create index id1 on parcels (id1);
create index id2 on parcels (id2);
create index id2DevType on parcels (id2,developmenttypecode);
create table DevelopmentTypes (GridCode CHAR, DevelopmentTypeName
TINYTEXT);
.mode tab
.import ‘C:/PECASProject/BaseYear/DevelopmentTypes.txt' DevelopmentTypes
.exit
```

5: SQLite script file to load in the text version of the database to the sqltiteparcels.db database as a part of SD module run orchestration: The file is under the sub directory ‘C:/PECAS*Project*/BaseYear’

sqliteBulkLoadBaseYear.sql

The specific script is as follows:

```
.mode tab
```

```
delete from parcels;
vacuum;
.import 'C:/PECASProject/BaseYear/BaseYearParcelDatabase.tsv' parcels
.exit
```

6: SQLite script file to load in the text version of the database in subsequent years to the sqltiteparcels.db database as a part of SD module run orchestration: The file is under the sub directory 'C:/PECAS*Project*/S01/AllYears/Code'

sqliteBulkLoadAllYears.sql

The specific script is as follows:

```
.mode tab
delete from parcels;
vacuum;
.import 'C:/PECASProject/S01/AllYears/Working/parceltxtout.tsv' parcels
.exit
```

7: A blank file under the subdirectory 'C:/PECASProject/S01/AllYears/Inputs'. This is required in order to specify what occurs in the logging process and in terms of what is sent to the screen and what is sent to the log file.

log4j.xml

8: The run controls file for the SD Module under the subdirectory 'C:/PECAS*Project*/S01AllYears/Inputs':

sd.properties

This file contains specifications regarding aspects of the algorithms the SD Module is to perform and the locations of certain inputs and outputs it is to use in this 'full' set-up in particular. The values for certain model parameters are also set in this file. The full set of variables and possible settings for this file are described in section 5.3 in this document. Note that the properties are common for all the years thus is places in the 'C:/PECAS*Project*/S01AllYears/Inputs' folder.

The specific settings required in this 'full' set-up are as follows:

```
LandJDBCDriver=org.sqlite.JDBC
InputJDBCDriver=sun.jdbc.odbc.JdbcOdbcDriver
LandDatabase=jdbc:sqlite:C:/PECASProject/Csd/AllYears/Working/sqliteparcels.db
InputDatabase=jdbc:odbc:ProjectPECASInputsS01
ExcelInputDatabase=true
ExcelLandDatabase=false
LandTable=parcels

#*******************These are for operation *****************
UseSQLInputs=true
UseSQLParcels=true
DevelopmentDispersionParameter=.0000005
DevelopmentAlternativesDispersionParameter=0.0000005
IntensityDispersionParameter=.000001
LUZColumnName=ID3
TAZColumnName=ID2
CoverageColumnName=DevelopmentTypeCode
QuantityColumnName=AmountOfDevelopment
ZoningColumnName=ZoningSchemeCode
```

```
SizeColumnName=AmountOfLand
YearBuiltColumnName=YearBuilt
AAResultsDirectory= C:/PECASProject/S01/
SDOutputsDirectory= C:/PECASProject/S01/AllYears/Outputs/
LogFilePath= C:/PECASProject/S01/
IntegerCodeForCoverage=true
MaxParcelSize=1
InputParcelFile=C:\\PECASProject/\\S01\\AllYears\\Code\\sqlitebulkloadAllYears.sql
OutputParcelFile= C:/PECASProject/Csd/AllYears/Working/parcelstxtout.tsv
```

9: The batch files containing the commands for running the model in a ‘full‘ set-up in the sub-directory ‘C:\PECAS*Project*\S01’:

RunS01.cmd

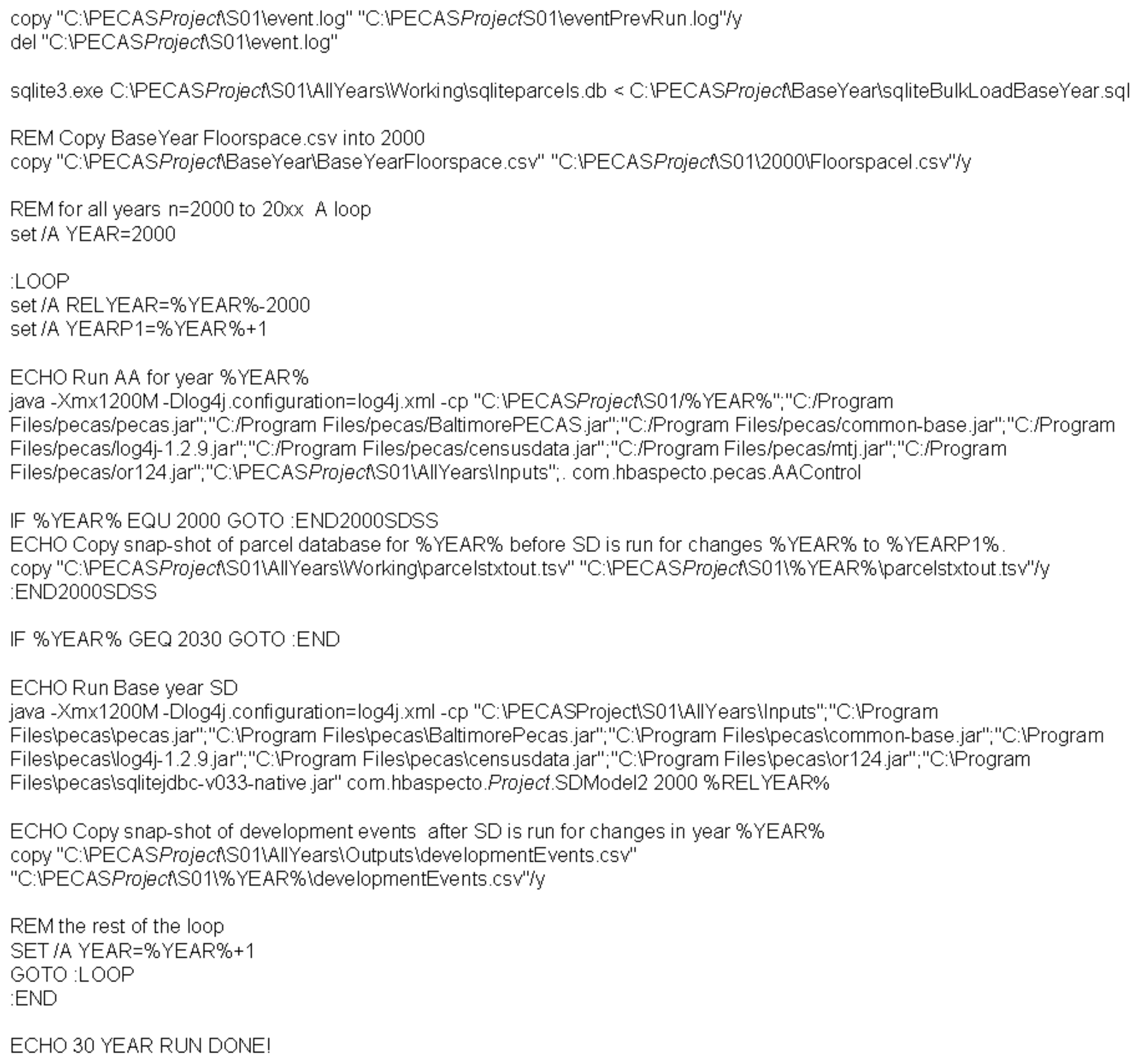
With this batch files in place, the AA Module and SD module is run through the years (in a ‘full’ set-up in this sub-directory) by entering ‘RunS01’ a DOS command window as described in section 6.4.

The specific contents of the batch file for a 30 year run from 2000 to 2030 is listed below in figure 6. The purpose of each line and what it does is covered under run orchestration in the section 6.5

6.4 Operating System Settings

The settings required for the Windows operating system to make the appropriate connections between the PECAS Modules and their input and output files are indicated below.

Again, in these indications, the italicized text characters *'Project'* are used to indicate a text description of the specific model project or area being considered.



```
copy "C:\PECASProject\S01\event.log" "C:\PECASProjectS01\eventPrevRun.log"/y
del "C:\PECASProject\S01\event.log"

sqlite3.exe C:\PECASProject\S01\AllYears\Working\sqliteparcels.db < C:\PECASProject\BaseYear\sqliteBulkLoadBaseYear.sql

REM Copy BaseYear Floorspace.csv into 2000
copy "C:\PECASProject\BaseYear\BaseYearFloorspace.csv" "C:\PECASProject\S01\2000\FloorspaceI.csv"/y

REM for all years n=2000 to 20xx  A loop
set /A YEAR=2000

:LOOP
set /A RELYEAR=%YEAR%-2000
set /A YEARP1=%YEAR%+1

ECHO Run AA for year %YEAR%
java -Xmx1200M -Dlog4j.configuration=log4j.xml -cp "C:\PECASProject\S01/%YEAR%";"C:/Program
Files/pecas/pecas.jar";"C:/Program Files/pecas/BaltimorePECAS.jar";"C:/Program Files/pecas/common-base.jar";"C:/Program
Files/pecas/log4j-1.2.9.jar";"C:/Program Files/pecas/censusdata.jar";"C:/Program Files/pecas/mtj.jar";"C:/Program
Files/pecas/or124.jar";"C:\PECASProject\S01\AllYears\Inputs";. com.hbaspecto.pecas.AAControl

IF %YEAR% EQU 2000 GOTO :END2000SDSS
ECHO Copy snap-shot of parcel database for %YEAR% before SD is run for changes %YEAR% to %YEARP1%.
copy "C:\PECASProject\S01\AllYears\Working\parcelstxtout.tsv" "C:\PECASProject\S01\%YEAR%\parcelstxtout.tsv"/y
:END2000SDSS

IF %YEAR% GEQ 2030 GOTO :END

ECHO Run Base year SD
java -Xmx1200M -Dlog4j.configuration=log4j.xml -cp "C:\PECASProject\S01\AllYears\Inputs";"C:\Program
Files\pecas\pecas.jar";"C:\Program Files\pecas\BaltimorePecas.jar";"C:\Program Files\pecas\common-base.jar";"C:\Program
Files\pecas\log4j-1.2.9.jar";"C:\Program Files\pecas\censusdata.jar";"C:\Program Files\pecas\or124.jar";"C:\Program
Files\pecas\sqlitejdbc-v033-native.jar" com.hbaspecto.Project.SDModel2 2000 %RELYEAR%

ECHO Copy snap-shot of development events  after SD is run for changes in year %YEAR%
copy "C:\PECASProject\S01\AllYears\Outputs\developmentEvents.csv"
"C:\PECASProject\S01\%YEAR%\developmentEvents.csv"/y

REM the rest of the loop
SET /A YEAR=%YEAR%+1
GOTO :LOOP
:END

ECHO 30 YEAR RUN DONE!
```

Figure 6: Script of 'runS01.cmd' file to run AA and SD module through time

ODBC Data Source Links

The ODBC settings must be established for the windows operating system so that it knows that a specific driver (excel for this program) is used as a data source.

There will be four links to be established for the following input files:

*Project*PECASInputs.xls
*Project*PECASInputsSepAA.xls
*Project*PECASInputsSepSD.xls
*Project*PECASInputsFullAASD.xls

Following instructions are for establishing the link for input file for the *Project*PECASInputs.xls for a run of the full model system through time (using the 'full' set-up and scenario S01)

Go to: Start > Control Panel > Administrative Tools > Data Sources (ODBC)

1: Select Systems DSN tab: by selecting systems DSN, every user is able to access this database. The System DSN can only be accessed with administrative privilege. Otherwise 'User DSN' can be selected but the process has to be repeated for all the users of pecas system.
2: Add a driver for Microsoft Excel. (select driver do Microsoft Excel[*.xls])
3: Type Datasource name as '*Project*PECASInputs'
4: Select 'Workbook" and browse to '*Project*PECASInputs.xls' file in 'C:\PECAS*Project*\S01\AllYears\Inputs'
5: Press 'Ok'

Path Environment Variable

This process is done only if the SQLite files sqlite.exe and sqlite3.dll can NOT be placed in the directory 'C:\Windows\system32' due to administrative privileges. The Path specification needs to be extended to include the sub-directory location of the SQLite files, as follows:

Go to: Start > Control Panel > System > Advanced > Environment Variables

1: Edit the Path value by appending to the end of it 'C:\Program Files\PECAS*Project*'

DOS Command Windows

A DOS Command window shortcut must be created to run the PECAS model. The steps of creating such shortcut and the changes to the properties are discussed here;

1: Create a DOS command shortcut on the desktop.

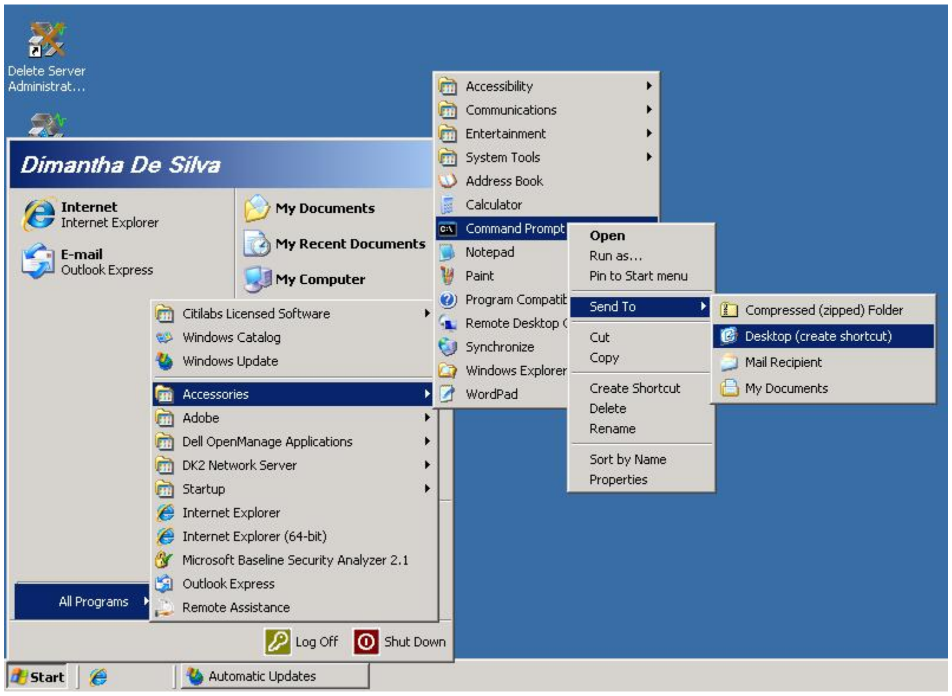
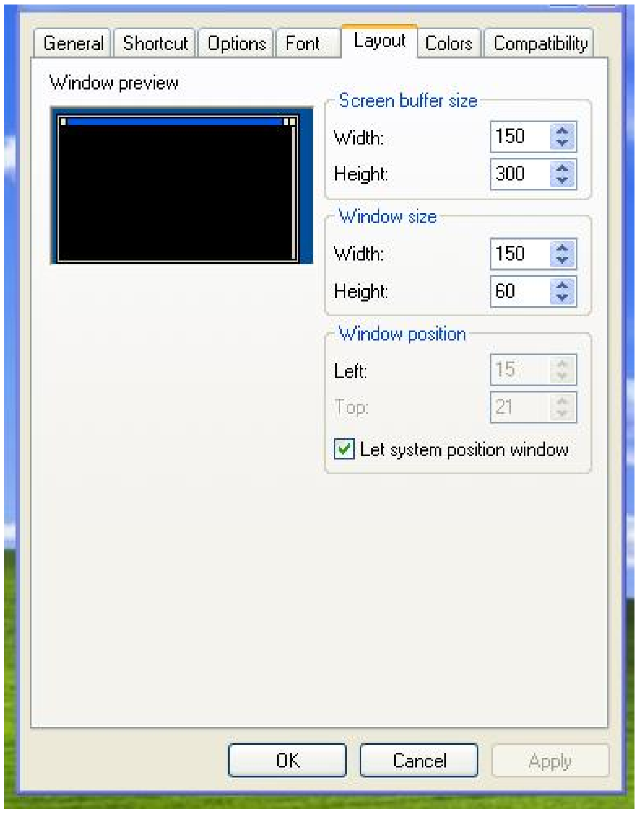


Figure 7: Screen shot to create a DOS Command window

2: Rename the shortcut to be *Project*PECASxxxxxx where xxxxxx is a appropriate name for the scenario or calibration considered. That is 'AACalibration' for stage two AA calibration, 'SD Calibration' for stage two SD calibration and 'S01' for a scenario 1 of 'full' set-up of application.

3: Change the properties of the DOS command window by right clicking on the shortcut and selecting 'properties'.

Go to the 'Layout' tab and changes the values as in Figure 8. These changes are done so that the Command window is made to the full size of the screen so that the commands and run full information is shown during the PECAS run.

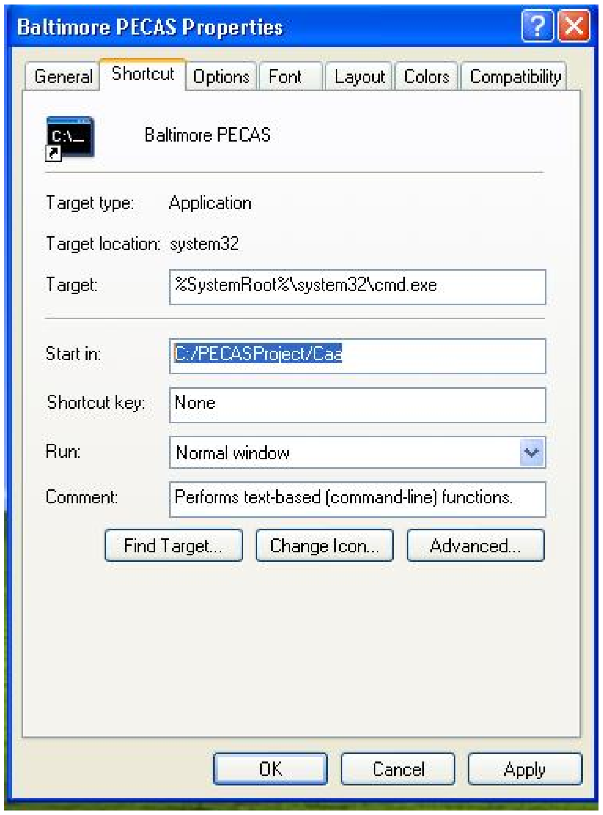


Screen buffer size
Width: 150
Height: 300

Window size
Width: 150
Height: 60

Window position
Left: 15
Top: 21

☑ Let system position window



Baltimore PECAS

Target type: Application
Target location: system32
Target: %SystemRoot%\system32\cmd.exe
Start in: C:/PECASProject/Caa
Shortcut key: None
Run: Normal window
Comment: Performs text-based (command-line) functions.

Figure 8: Screen shot of changes to properties of the DOS Command Window

Go to ‘Shortcut’ tab and change the ‘start in:’ to be ‘C:\PECAS*Project*\Caa’ for stage two AA calibration, ‘C:\PECAS*Project*\Csd’ for stage two SD calibration, and ‘C:\PECAS*Project*\S01’ for scenario ‘S01’ for a ‘full’ set-up.

5: Press ‘Apply’

6.5 Run Orchestration

The run orchestration describes the process of running the model system through time. (What happens when ‘runS01’ is typed on the DOS Command window)

The options for ‘Full’ Set-up run orchestration available are;

(a) Windows Batch File
(b) Cube Voyager

Each of these is described below.

Windows Batch File

The process shown here is run using a DOS command file ‘runS01.cmd’ stored in the directory ‘C:\PECAS*Project*\S01’shown in figure 6. The scenario considered here is S01 running AA and SD module through time from Base year 2000 to the year 2030. Snap shot of the parcel database and the development events occurring in SD module are taken every year.

The schematic diagram of the run orchestration is shown in figure 9. Each element is is described below.

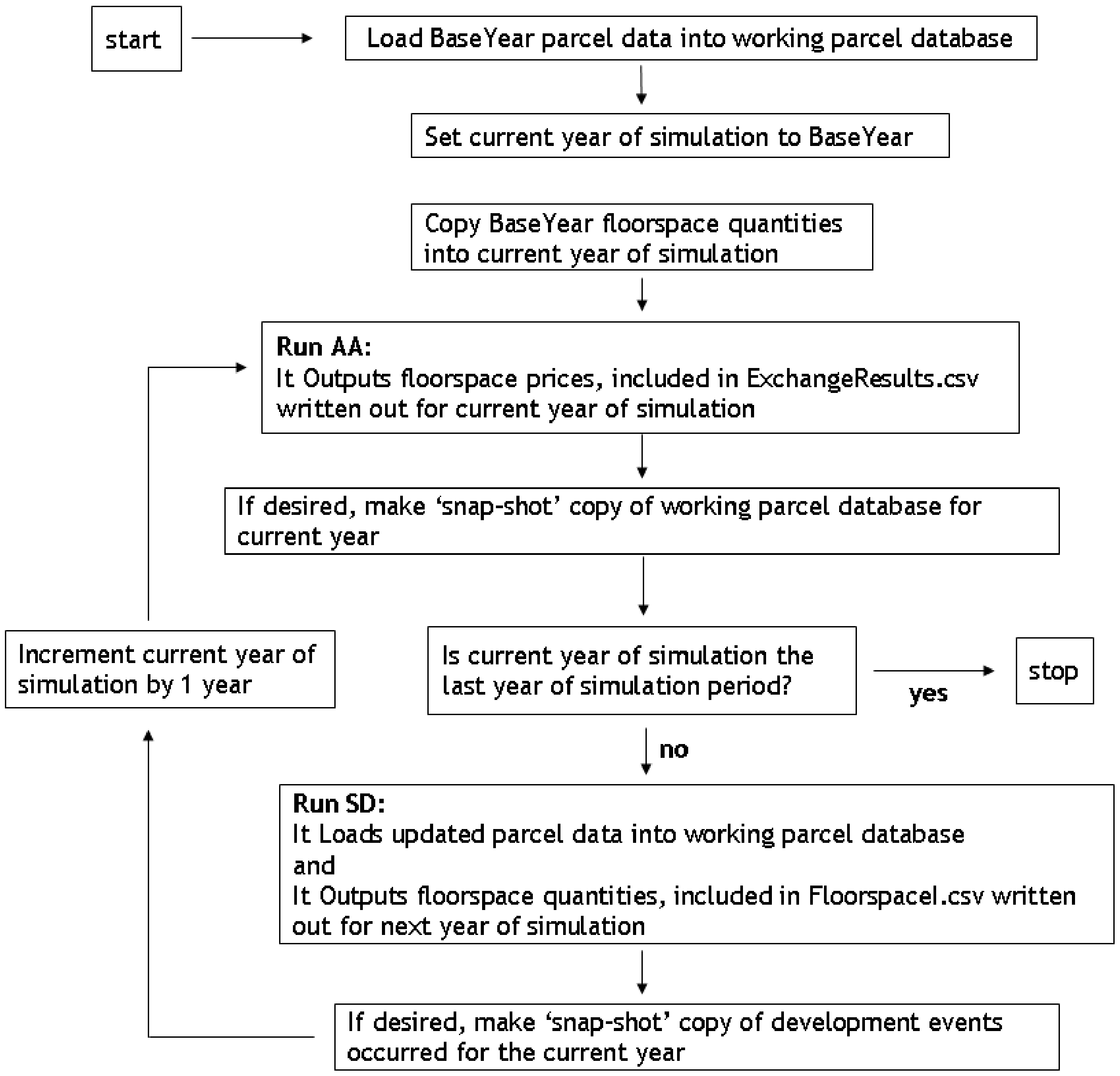

start
Load BaseYear parcel data into working parcel database
Set current year of simulation to BaseYear
Copy BaseYear floorspace quantities into current year of simulation
Run AA:
It Outputs floorspace prices, included in ExchangeResults.csv written out for current year of simulation
If desired, make 'snap-shot' copy of working parcel database for current year
Increment current year of simulation by 1 year
Is current year of simulation the last year of simulation period?
yes
stop
no
Run SD:
It Loads updated parcel data into working parcel database and It Outputs floorspace quantities, included in FloorspaceI.csv written out for next year of simulation
If desired, make 'snap-shot' copy of development events occurred for the current year

Figure 9: Schematic diagram of run Orchestration using Windows batch file

Start

The scenario run is started by typing 'RunS01' on a DOS command window created as in section 6.4

__Load BaseYear parcel data into working parcel database;__

The text version of the base year parcel database, baseyearparcelstxt.tsv is loaded in to the SQLite database sqliteparcels.db by a script file squliteBulkLoadBaseYear.sql. The content and location of each of these files are described under section 6.3

The command line statement in the file 'RunS01.cmd' shown in Figure 6 is of the form:

*Sqlite3.exe C:\PECAS\S01\AllYears\Working\sqliteparcels.db <
C:\PECASProject\S01\BaseYear\sqliteBulkLoadBaseYear.sql*

__Set current year of simulation to BaseYear__

The model system is run for each year in the span of years starting at the base year (year 2000 in this case) and ending at the Last Year (year 2030 in this case). The value of the current year being considered at any point in this run process is stored in the DOS variable 'YEAR' in the file 'RunS01.cmd'. At this point, at the start of the simulation process, the value for 'year' is set to base year.

The command line statement in the file 'RunS01.cmd' is of the form:

set /A YEAR=2000

__Copy Base Year floorspace quantities into current year of simulation__

The space quantities for each type of space in each LUZ for the BaseYear of the simulation are copied into the directory for the current year of the simulation (which is the Year 2000 at this point) as required input for AA module. These space quantities for the BaseYear are in the file 'BaseYearFloorspace.csv' stored in the 'BaseYear' subdirectory.

The command line statement in the file ‘RunS01.cmd’ is of the form:

*copy "C:\PECASProject\BaseYear\BaseYearFloorspace.csv"
"C:\PECASProject\S01\2000\FloorspaceI.csv"/y*

Run AA:

A loop is created at this point to run the model from 2000 to 2030. Two variables for the relative year from the base year and the next year for the run are stored under DOS variable ‘RELYEAR’ and ‘YEARP1’ respectively.

The command line statement in the file ‘RunS01.cmd’

*:LOOP
set /A RELYEAR=%YEAR%-2000
set /A YEARP1=%YEAR%+1*

AA is run with a command line that tells it to use java with the relevant inputs (, these included in a long list of input parameters indicating the current year, relative year from the base year, computer memory to use and the locations for the program code, the input files, the output files and the log report for the run.

The command line statement in the file ‘RunS01.cmd’ is of the form:

*java -Xmx1200M -Dlog4j.configuration=log4j.xml -cp "C:\PECASProject\S01/%YEAR%";
"C:/Program Files/pecas/pecas.jar";"C:/Program Files/pecas/ProjectPECAS.jar";
"C:/Program Files/pecas/common-base.jar";"C:/Program Files/pecas/log4j-1.2.9.jar";
"C:/Program Files/pecas/censusdata.jar"; "C:/Program Files/pecas/mtj.jar"; "C:/Program*

Files/pecas/or124.jar";"C:\PECASProject\S01\AllYears\Inputs";.com.hbaspecto.pecas.AAControl

The information provided specifically in the order it is stated in the above script are (1) the amount of computer memory to use is 1200MB; (2) the subdirectory containing the aa.properties file is in 'C:\PECAS*Project*\S01\%YEAR%\', this aa.properties file indicates the subdirectories where AA is to read its input files including FloorspaceI.csv (containing the floor space quantities), ActivityTotalsI.csv (containing the total amount of activity by type) and Skims*Year.*csv (containing the transportation utilities). Also the subdirectories where AA is to write its output files including ExchangeResults.csv (containing the space price); (3) the location of necessary java source code is in the set of files, all stored in the subdirectory 'C:/Program Files/pecas/'; (4) the file *Project*PECASInputs.xls containing the run controls is in the subdirectory 'C:\PECAS*Project*\S01\AllYears\Inputs'; (5) the java processor to use is com.hbaspecto.pecas.AAControl.

<u>If desired, make 'snap-shot' copy of working parcel database for current year</u>

At this point, a snap shot is taken of the working parcel database for current year before it is changes by running SD. This is not required for the base year, as it is already have been done by a previous step, thus a condition is given to ignore the command if the current year is 2000. Alternatively, the statement can be changed only to copy this file in the years that is needed.

The command line statement in the file 'RunS01.cmd' is of the form:

IF %YEAR% EQU 2000 GOTO :END2000SDSS
copy "C:\PECASProject\S01\AllYears\Working\parcelstxtout.tsv" "C:\PECASProject\S01\%YEAR%\parcelstxtout.tsv"/y

:END2000SDSS

<u>Is current year of simulation the last year of simulation period?</u>

The model system checks whether the current year of simulation is the last year of the period considered. If yes, it will stop the model and if not, it will continue to run SD module.

The command line statement in the file 'RunS01.cmd' is of the form:

IF %YEAR% GEQ 2030 GOTO :END

<u>Run SD:</u>

SD is run with a command line that tells it to use java with the relevant inputs, these included in a long list of input parameters indicating the current year, relative year from the base year, computer memory to use and the locations for the program code, the input files, the output files and the log report for the run.

The command line statement in the file 'RunS01.cmd' is of the form:

```
java -Xmx1200M -Dlog4j.configuration=log4j.xml –cp "C:\PECASProject\S01\AllYears\
Inputs"; "C:\Program Files\pecas\pecas.jar"; "C:\Program Files\pecas\ProjectPecas.jar"
;"C:\Program Files\pecas\common-base.jar";"C:\Program Files\pecas\log4j-1.2.9.jar";
"C:\Program Files\pecas\censusdata.jar";"C:\Program Files\pecas\or124.jar";
"C:\Program Files\pecas\sqlitejdbc-v033-native.jar" com.hbaspecto.Project.SDModel2
2000 %RELYEAR%
```

The information provided specifically in the order it is stated in the above script are (1) the amount of computer memory to use is 1200MB; (2) the subdirectory containing the sd.properties file is in 'C:\PECAS*Project*\S01\Allyears\Inputs\', this sd.properties file indicates the subdirectories where SD is to read its input files including ExchangeResults.csv (containing the space prices) and the subdirectories where AA is to write its output files including FloorspaceI.csv (containing the space quantity) to be written to the next year of simulation; (3) the location of necessary java source code is in the set of files, all stored in the subdirectory 'C:/Program Files/pecas/'; (4) the file *Project*PECASInputs.xls containing the run controls is in the subdirectory 'C:\PECAS*Project*\S01\AllYears\Inputs'; (5) the java processor to use is com.hbaspecto.Project.SDModel2 and (6) the number of relative years from the base year defined by 'RELYEAR'

<u>If desired, make 'snap-shot' copy of development events occurred for the current year</u>

At this point, a snap shot is taken of the development events occurred for current year before running AA for the next year and stored in the year folder in simulation.

The command line statement in the file 'RunS01.cmd' is of the form:

copy "C:\PECASProject\S01\AllYears\Outputs\developmentEvents.csv"
"C:\PECASProject\S01\%YEAR%\developmentEvents.csv"/y

<u>Increment current year of simulation by 1 year</u>

The model system increment the current year of simulation by 1 year and direct the model to the start of the loop.

The command line statement in the file 'RunS01.cmd' is of the form:

```
SET /A YEAR=%YEAR%+1
GOTO :LOOP
```

6.6 Software License and Copyright Matters

6.7 Acknowledgements – may put this elsewhere, just a ‘note-holder’ spot now

- framework developed over a series of projects by JD Hunt and JE Abraham of HBA Specto, including: Oregon, Ohio, Sacramento, Baltimore, Edmonton, California, Mumbai
- portions of code established and modified in an evolution over these projects, primarily by JE Abraham but with multiple contributions by others;
- projects in Oregon and in Ohio does as sub in teams led by PB; several members of these teams made substantial contributions to the code, including Christi Willisden, Joel Freedman and Chris Frazer?) of Parsons Brinckerhoff and Tim Heier.
- Oregon also funded work in the development of their statewide model and adopted the apache license, etc
- acknowledge California funding

- acknowledge IAPR support

7. variables index

correspondence tables for variables in different documents, taking tables from Theoretical Formulation document and putting them together here, with several versions sorted several ways

8. Interpreting Error Messages and Troubleshooting