SOFEA© (SOil Fumigant Exposure Assessment system)

Programmer's Guide

Steven A. Cryer
August 2004

sacryer@dow.com

Dow AgroSciences
9330 Zionsville Road
Indianapolis, IN 46268

Abstract

The Soil Fumigant Exposure Assessment (SOFEA©) system was developed to address exposure concerns associated with the use of soil fumigants. The USEPA model ISCST3 forms the basis for the numerical system. SOFEA automates the generation of input files, adds a Monte Carlo component to ISCST3, and summarizes output results from ISCST3 in a systematic way. SOFEA is largely written in Visual Basic for Applications (VBA) using Microsoft Excel as the core program, although a specific optimization program was written in FORTRAN 90. ISCST3 and the optimization program are called from VBA and results are brought back into VBA to make the transition transparent to the user. This manuscript serves as a programmers guide for use by experienced VBA and FORTRAN programmers who may require modification of the source code to enhance or supplement the current attributes of SOFEA. Descriptions of the algorithms and the subroutines that execute them are summarized.

Introduction

SOFEA is a powerful and useful pre and post processor for ISCST3 when considering exposure issues surrounding soil fumigants (Cryer, 2004). ISCST3 is a regulatory air dispersion model developed by the USEPA that uses Gaussian plume approximations for transport predictions (ISCST3, 1995). SOFEA provides the interface to generate complex ISCST3 input files that can contain multiple transient source terms of different strengths and spatial/temporal attributes as dictated by agronomic practices for the region being simulated. PDFs can be continuous (i.e., normal, triangular …), or discrete based upon experimental/field data by using the fitting tools found in the Excel add-in software package Crystal Ball 2000 Pro (Trademark of Decisioneering, Inc.). Coupling Excel, Crystal Ball 2000 Pro, and ISCST3, along with a stand-alone program for optimization (discussed later), with VBA, allows the source code for deterministic models to remain unmodified for stochastic implementation. Often, regulatory constraints dictate the model source code cannot be altered if simulation results are to be used for registration or regulatory purposes.

Methods and Materials

Monte Carlo/Latin Hypercube Implementation

Parameters required by ISCST3 input (and SOFEA) are predominately found in the worksheet "PDF_Parameters". SOFEA reads in the user specified input parameters and generates ISCST3 input files. Parameters can be single valued or assigned probability Density Functions (PDFs) to account for parametric uncertainty/variability for regional assessments. The choice for continuous PDF distribution type are lognormal, weibull, gamma, exponential, pareto, extreme value, beta, logistic, normal, triangle, and uniform. Current parameters that can be assigned PDFs are given in Table 1. Cells in Microsoft Excel can be assigned PDFs and these PDFs are sampled using Monte Carlo and/or Latin Hypercube sampling by using the third party add-in software package Crystal Ball 2000 Pro (Trademark of Decisioneering, Inc.). Crystal Ball allows all spreadsheet cells to be expressed as probability density functions for Monte Carlo simulation. Monte Carlo PDFs are sampled for at least 400 iterations to generate (in memory) field attributes for internal use within SOFEA.

Table 1. Parameters in SOFEA that can be assigned a PDF

Crop % (up to 5 crops allowed)	Application rate per crop	Application date per crop
Field size per crop	% Drip Applications per crop	Incorporation depth per crop
Tarp coverage per crop	Weather year	

GIS Raster Approximations

SOFEA has the capability for data input from GIS data bases that can include crop cover, elevation and population. The original design for SOFEA was to allow the capability of hand entry of GIS information from hard copy maps. Thus, a single township (6 mi x 6 mi or 9565 m x 9656 m) was divided into a 10x10 raster based grid (Figure 1). Each grid in the 10x10 township (6 mi x 6 mi) is assigned specific data via worksheets "LandCover", "Population" and "Elevation". In addition, the worksheet "GIS Data" can contain direct data dumps from GIS

software for use with SOFEA. Table 2 summarizes the SOFEA worksheets that require populated via GIS (or user selected) information.

Table 2. Worksheets of SOFEA containing GIS information or where GIS information can be entered.

Worksheet	Description
GIS_data	Contains raw data for township for the 10x10 raster based grid system
LandCover	Worksheet where user can hand enter GIS information for land cover type (ag-capable, urban, water) or read this information from the worksheet "GIS_data"
Population	Worksheet where user can hand enter GIS information for population or read this information from the worksheet "GIS_data"
Elevation	Worksheet where user can hand enter GIS information for elevation or read this information from the worksheet "GIS_data"

There are buttons on each worksheet where the user can reset each grid to zero, or use information using GIS software/databases (such as ArcView). Macros assigned to buttons in these worksheets are summarized in Table 3.

Table 3. Worksheet where buttons linked to macros perform GIS driven algorithms.

Worksheet – Macro	Description
LandCover	
Land_null	Sets all land to ag-capable (= 0)
GIS_data_for_LandCover	Gets GIS information from the worksheet GIS_data, populates cells and generates land cover graphic in worksheet "LandCover"
LandCover_update	Uses the user specified cell information in "LandCover" and generates land cover contour graphic
Elevation	
Elev_null	Sets all elevations to zero (i.e., flat)
GIS_data_for_Elevation	Gets GIS information from the worksheet GIS_data, populates cells and generates elevation graphic in worksheet "Elevation"
Elevation_update	Uses the user specified cell information in "Elevation" and generates elevation contour graphic
Population	
Pop_null	Sets all population density to zero
GIS_data_for_Population	Gets GIS information from the worksheet GIS_data, populates cells and generates population contour graphic in worksheet "Population"
Population_update	Uses the user specified cell information in "Population" and generates population contour graphic

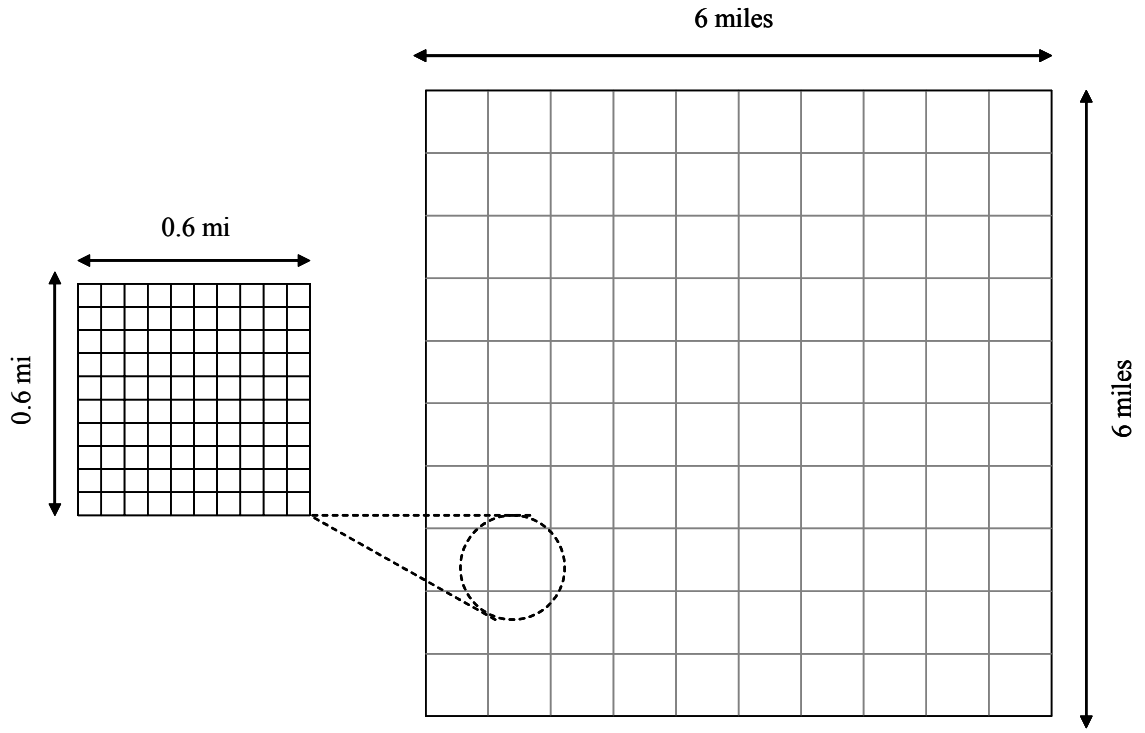

Figure 1. Uniform grid raster for a single township. Coarse grid illustrates user supplied scale, while refined grid is what is used internal to SOFEA.

The internal raster based grid for SOFEA for each township is 100 x 100, where each of the "refined" grid has dimensions of 96.56 x 95.56. This value of grid dimensions is listed as a parameter (Cell E17 in worksheet "PDF_Parameters"). A typical user shouldn't adjust this value since it relates to the "resolution" of the GIS data. This value is used by SOFEA, but for rounding and estimating [i.e., some times the GIS data dumps for a township have a grid of 12x10 (for example)]. Thus, the model interpolates via knowing the size of the grid we are after is 96.56 x 96.56 m (10 x 10).

GIS information has been discretized such that a township can (and currently is) represented by a coarse 10x10 raster based grid. Thus, the length of a township divided by 10 gives the 965.6 meter grid size. So when a user manually types in GIS information in worksheets such as "Land Cover" (i.e. by putting a 0, 1, 2, or 3 in an appropriate cell), this cell represents a 969.6 m x 965.6 m chunk of real estate in the township that has unique properties (in terms of land cover, elevation, and population). This chunk of land is further subdivided (internally) into a 10x10

grid, but each of the new elements of the grid are assigned the same value as found in the coarser grid. Thus, even though internal dimensions for the refined grid are 96.56 m x 96.56 m, the data used to generated the refined grid is based on the coarse grid of 10x10 for a township. Typically, GIS information is more detailed than the 10x10 township grid. Thus, detailed data must be made "coarser" to yield data for the 10x10 raster grid township. The coarse GIS data is entered in the worksheet "GIS_data". Subroutines that read in the coarse GIS information from worksheets "LandCover", "Elevation", and "Population" are GIS_data_for_LandCover, GIS_data_for_Elevation, and GIS_data_for_Population, respectively.

Modifications for finer GIS resolution

GIS information is used to determine field placement (ag-capable land), topography (terrain input for ISCST3), and population densities for use in risk assessment. There may be cases where a 965.6 m x 965.6 m grid is two large to resolve the detail the user is after. An example would be a large apartment complex in the middle of a rural area. A high population density is captured in a small spatial area. This type of detail may be available in the original GIS data, but when averaging this information to obtain a 10x10 township values, this detail is lost.

If higher resolution is required, a programmer can easily use a grid system of 100x100 for each township for reading in GIS data. Modifications can be made to the subroutines LandCover_color, Terrain_elev, pop_census to read in 100x100 resolution township data (from an ASCII file or another worksheet) for land cover, elevation, and population, respectively. It is in these subroutines where the 10x10 grid system using worksheets found in Table 2 are further subdivided into the smaller raster based grid (to give a grid system of 100x100 for each township). This subroutine can be used to read in information for a 100x100 township raster grid in lieu of reading in coarse data.

The arrays where GIS information is stored internal to SOFEA are

Location(k, i, j) = Land Cover type at (i,j) location for township k.
Elevation(k, i, j) = Elevation at (i,j) location for township k

Population(k, i, j) = Population at (i,j) location for township k

Where

k = township ID ($1 \leq k \leq 9$)
i = row within refined grid system ($1 \leq i \leq 100$)
j = column within refined grid system ($1 \leq j \leq 100$).

Source Strength (Transient Flux of Fumigant)

Mass volatilization loss for soil fumigants from soil can be estimated by field measurements or numerical predictions. Predicted volatility loss from soil can be specified as transient source terms for air dispersion modeling.

Application Scaling Factor

Measured flux rates, specific for the conditions at the time of the field study, are adjusted and used as model source terms. The adjustments for volatility losses are based upon depth of incorporation and time of year in an attempt to represent the complete flux response surface (Cryer, 2004). It is assumed that flux rate is proportional to the application rate. Two different experimental or numerically generated flux profiles can be used. For assessments performed by van Wesenbeeck *et al*. (2004), drip and shank flux profiles based upon actual field studies performed in California were used. If a surface drip application was selected, then the field observation for surface drip was used and scaled appropriately, etc. The transient flux loss used in the simulations for each field is given by Eq. 1.

$$Flux_i = \frac{R}{R_{ref}} * F_i * S_{incorp} * S_{yr} \quad = \quad Fscale * \frac{F_i}{R_{ref}} \qquad (1)$$

$Flux_i$ = Appropriately scaled hourly flux loss for hour "i" based upon observations of field trial
R = pesticide application rate (kg/ha) obtained by sampling user defined PDF
F_i = experimentally observed flux rate (reference profile) for hour "i"
R_{ref} = pesticide application rate (kg/ha) for the experimentally observed flux profile

S_{incorp} = scaling factor for depth of incorporation (dimensionless)
S_{yr} = scaling factor for time of year (dimensionless).
Fscale = R * S_{incorp} * S_{yr}

The scaling factors S_{incorp} and S_{yr} are discussed in the following sections.

SOFEA reads in F_i from the worksheet "Flux_files" via the subroutine "Get_flux". Units for flux are in g m^{-2} s^{-1}. This flux is scaled (divided) by the reference application rate (kg/ha) such that upon multiplication by the PDF sampled application rate (kg/ha). The units for flux required by ISCST3 input are achieved.

F_scale is defined in the subroutine "Fld_Placement".

Temporal (S_{yr})

Temporal scaling was broken down into a warm or cool season to account for the greater potential mass loss during warm seasons. The scaling of cumulative mass loss between cool (Sep 22 – Jun 21) and warm (Jun 22 – Sept 21) season emission rates was assigned a factor of 1.6 [B. Johnson California Department of Pesticide Regulations (CDPR), personal communication, 2001] for California use scenarios. If the reference field study was conducted in the winter, but for simulation purposes, a summer application was assumed, then the experimental winter flux loss was scaled by 1.6. This indirectly accounts for gross temperature effects for fumigant soil volatility losses from soil for a two-temperature regime year. S_{yr} is defined in the subroutine "Depth_Incorp".

Depth of incorporation (S_{incorp})

One approach to estimate the incorporation-scaling factor is to assume linearity with depth. Here, 100% mass loss is assumed for surface applications if no plastic tarp is present at the soil surface. If a tarp is present, then a default value of 64% of applied is assumed unless otherwise specified by the user. The value of 64% loss is based upon simulation results of PRZM3 where PRZM3 was modified to include the necessary surface boundary condition when a tarp is present (Cryer and van Wesenbeeck, 2001). The 64% of applied loss with tarp at the soil surface is also

consistent with small scale field measurements of 66% for shallow drip with tarp reported elsewhere (Wang et al. 2001). However, the 64% loss for surface applications can be overridden by the user if appropriate. The lower bound for the linear scaling is based upon the experimental observation of field losses, where the depth of incorporation is known.

A limitation for linear scaling is that at some finite value for incorporation depth (critical depth), volatilization will cease and/or goes negative. The critical depth is below the reference depth of the field study. Negative predictions for volatility loss (non-physical) can be set to zero as this critical depth is exceeded. An alternative approach is to assume that for any incorporation depth greater than the experimental reference value, the flux loss equals the experimental value. For example, if the field study incorporation depth was 18 inches, and the volatilization loss was 25 % of applied, than any incorporation depth > 18 inches will likewise result in a cumulative volatilization loss of 25%. Both approaches are unsatisfactory since they introduce unrealistic bias and are not continuous with depth, although the user can always assume non-linear scaling with depth. However, the latter approach has been implemented for linear scaling (i.e., flux loss remains constant at the reference rate for all depths greater than the reference depth). Figure 2 represents the qualitative behavior for flux loss with depth assuming linear and non-linear scaling, and if a tarp is present at the soil surface.

S_{incorp} is defined in the subroutine "Depth_Incorp". The linear and non-linear equations used for S_{incorp} are given by Equations 2 and 5.

Linear Scaling

ApDepth <= Depth_ref

$$S_{incorp} = \frac{[\frac{Pct\_vol\_max - Pct\_vol\_ref}{0 - Depth\_ref}]ApDepth + Pct\_vol)\_max}{Pct\_vol\_ref} \qquad (2)$$

Where

Pct_vol_max = maximum percentage of applied that can be lost if application is made at the surface [input in worksheet "PDF_Parameters" cells B48 or C48 for drip or shank, respectively].

Pct_vol_ref = percentage of applied lost from reference field (or numerical) flux trial [input in worksheet "PDF_Parameters" cells B46 or C46 for drip or shank, respectively].

Depth_ref = depth of incorporation for reference flux trial [input in worksheet "PDF_Parameters" cells B44 or C44 for drip or shank, respectively].

ApDepth = depth of incorporation for current source term being evaluated

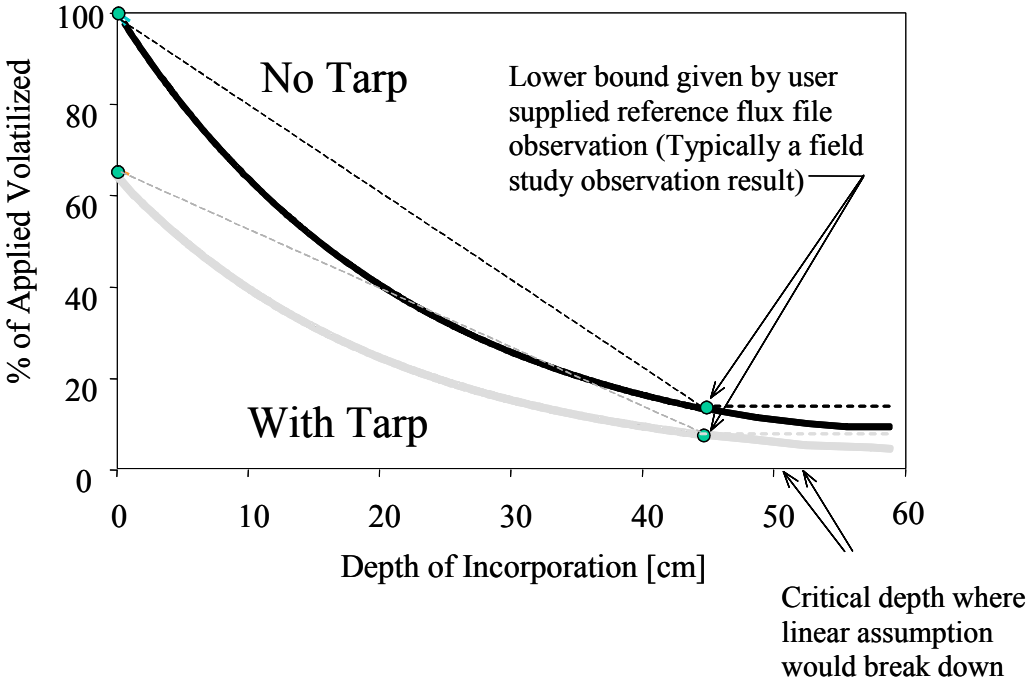

Figure 2. Qualitative representation of conservative nature of linear scaling with depth when numerical results indicate non-linear scaling is appropriate

Non-Linear Scaling

$$rate\_k = \frac{\ln\left(\frac{Pct\_vol\_ref}{Pct\_vol\_max}\right)}{Depth\_ref} \quad (3)$$

$$Pdepth = Pct\_vol\_max * Exp(rate\_k * ApDepth) \quad (4)$$

If Pdepth <= 0 Then Pdepth = 0
If Pdepth >= 100 Then Pdepth = 100

$$S_{incorp} = \frac{Pdepth}{Pct\_vol\_ref} \quad (5)$$

Source and Source Flux File Generation

Source files for each crop type for a given year of simulation are generated in the subroutine "Write_flux_files" that calls the subroutine "pop_fld_flux" to populate the array that summarizes the hourly emission rates for each field type for all hours within a single year. If an application date occurs within 15 days at the end of year, then the flux loss is "carried-over" to the beginning of the year (Note: end of year = 365*24 hours for non-leap year, or 366*24 hours for leap year).

All source file flux rates for every hour and every day for the year are written to the ISCST3 flux input files. Thus, there can be lots of zeros until the specific source application date is meet. Other sources may be emitting mass at this date, and the source ID and flux rate are captured in the flux input files for each hour of the simulation year.

Mass applied to specific area (Township)

SOFEA can have receptors in the central township of interest (1x1) or in neighboring townships to the central township of interest (3x3), but source terms can be placed within an air shed encompassing 23x23 townships. This is representative of a central township of interest surrounded by 11 townships on all sides (Figure 3). Township numbering is also represented in this figure.

A user supplied input parameter is the township allocation of a fumigant [kg]for townships within California (Cell E5 in worksheet "PDF_Parameters". This is a reference pesticide mass (M_{ref}) that can be applied to townships in the air shed being simulated. The reference (total) applied mass to a township is scaled up or down by a user defined scaling factor that is input by the user in worksheets "Twn_Mass_Wt" and "Twn_Mass_Wt_Ext", respectively, for each township in the simulation domain. The adjusted mass applied to a given township is simply the reference allocation amount multiplied by the scaling factor (Eq. 6).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92
93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115
116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138
139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161
162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184
185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230
231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253
254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276
277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299
300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322
323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345
346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368
369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391
392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414
415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437
438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460
461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483
484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506
507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529

Figure 3. Township numbering for a central township surrounding by 11 townships on all sides.

$$M_i = SF_i * M_{ref} \quad (6)$$

M_i = Fumigant mass [kg] applied to township i

SF_i = User supplied scale factor for township i [dimensionless]

M_{ref} = reference fumigant mass (use typical value or township allocation (if appropriate) – [kg]).

Thus, townships can receive more or less mass than the reference amount if the scaling factor is less than or greater than 1, respectively. A scale factor of zero indicates the township will receive no fumigant mass for the current year of simulation. Figure 4 represents example input

found in worksheet "Twn_Mass_Wt" where only the central 3x3 township information is input. The central 3x3 corresponds to the range of GIS information that can be used. Townships external to the central 3x3 can be allocated soil fumigant mass via the worksheet "Twn_Mass_Wt_Ext". However, the user needs to be aware that mass for the central 3x3 must only be input in one worksheet (either "Twn_Mass_Wt" or "Twn_Mass_Wt_Ext"). Thus, one worksheet will have all zeros for entries for the central 3x3. If this rule is not obeyed, then the mass allocated is the sum of that given by "Twn_Mass_Wt" and "Twn_Mass_Wt_Ext". In fact, all weights in worksheet "Twn_Mass_Wt" can be nulled out, the worksheet hidden, and all input can be made to the worksheet "Twn_Mass_Wt_Ext". These two worksheets exits since many users may not need to consider large areas for source placement (and visa versa).

0.216	0.066	0.09
0.316	1.040	0.148
0.479	0.70	0.00

Figure 4. Example of SOFEA input for 3x3 township allocation scaling factors for a reference allocation mass.

Weighting factors for the applied mass to a township are read into memory using the subroutines "fld_solve1" and "Read_Outside_3x3", where the latter subroutine is called by the former if the user has stated there will be fields outside the central 3x3 (Cell B112 in the worksheet "PDF_Parameters").

Field Placement (ISCST3 Sources)

Transient source terms are nematicide-treated fields where pesticide volatility can occur. Source strength and location are based upon experimental observations, management practices, agricultural capable land, and historical information. Much of the required data is geo-referenced and amenable to Geographic Information System (GIS) overlays and extraction.

Sources within a township can be placed randomly in agricultural land or weighted to specific township locations. Agriculturally- (ag) capable land is defined as all land excluding urban areas, water bodies, barren, rock, quarries, and wetlands. The total number of source terms selected is dependant on the total amount of pesticide mass allowed in a given township (i.e., the township allocation) and is a function of the field size, application rate, date, and depth of incorporation. Actual mass applied is the application rate (kg/ha) x field size (ha), while adjusted mass is the actual mass multiplied by CDPR scaling factors (dimensionless) as defined in Figure 5. The user specifies a reference township allocation for a simulation [kg]. The township allocation is the actual mass in the township for all soil fumigants other than 1,3-D and the adjusted mass for the soil fumigant 1,3-D only when in the state of California. Which application factors to use are given by the user choice for the integer flag "CA_13D_Scen" that is an input in the worksheet "PDF Parameters". No further source terms are allowed once the township allocation is met. In this way, the sensitivity of the township allocation to acute, sub-chronic, and chronic air concentrations can be addressed.

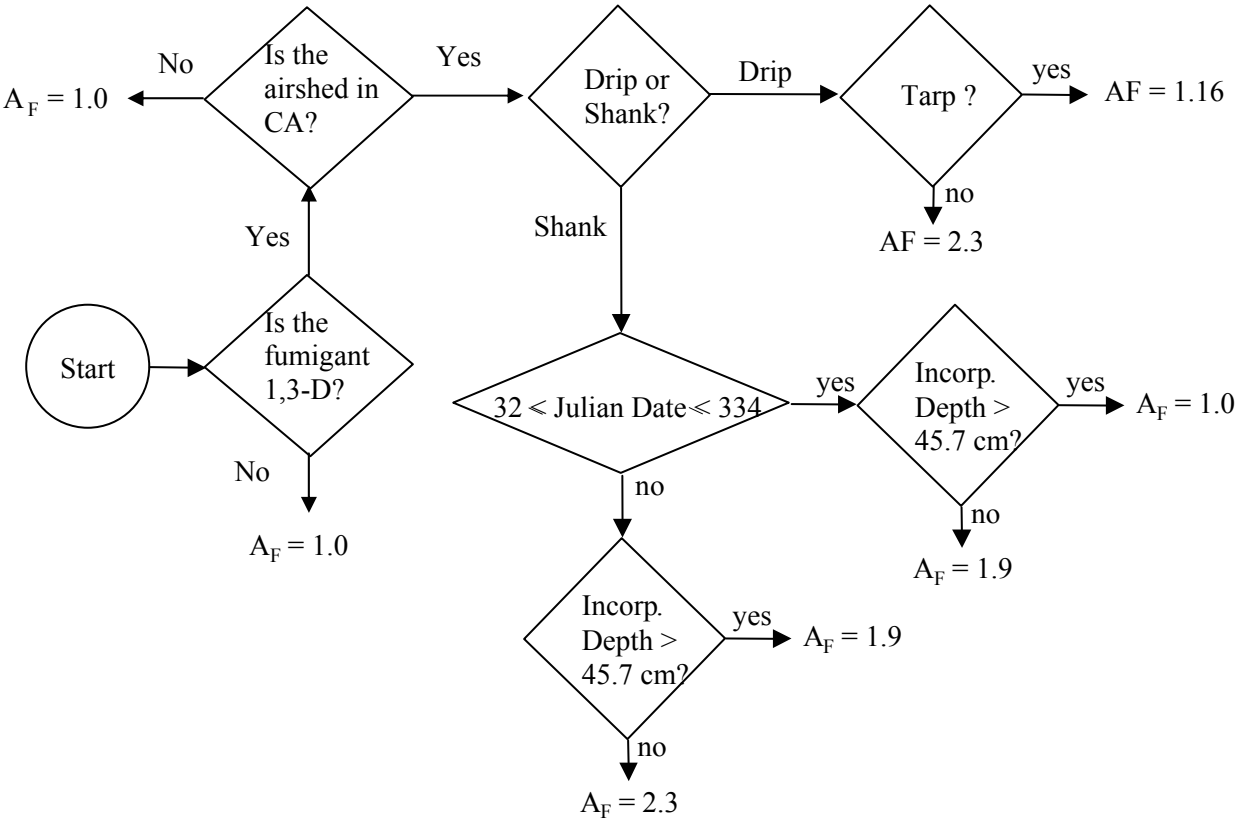

Figure 5. Methodology for determining the application factor for "correcting" the actual mass applied to a pseudo "Adjusted Mass". AF ≠ 1.0 are for 1,3-D in CA only.

Random Assignment

The constructed numerical system assumes a maximum of 100,000 iterations when attempting to randomly place fields within a township. Figure 6 illustrates that 100,000 iterations for random field placement are acceptable to adequately cover the entire township domain (where each point in Figure 5 represents the southwest corner of a treated field).

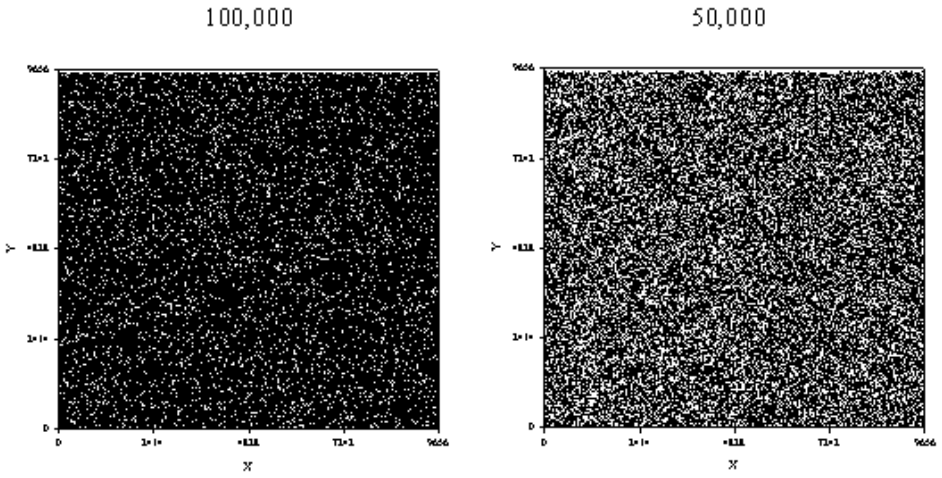

Figure 6. Example of a 100,000 and 50,000 randomly sampled SW field location for placement of treated fields within a township.

Section Weighted

For some Townships, certain sections within a township (1/36 township area) traditionally apply larger quantities of soil fumigant than do other township sections. In addition, it is possible that a farmer can repeatedly treat a field with the same soil fumigant for several consecutive growing seasons. Receptors in such sections, and near repeatedly treated fields will register higher chronic and possible acute fumigant air concentrations due to the spatial intensity of fumigant use. Simulations can be further refined to reflect repeated applications to the same fields.

Historical information is indicative of current and possibly future pesticide use patterns. In these situations, the more refined section grids of a township are used. The California numbering of township sections is given in Figure 7. For numerical implementation, the user specifies the probability of each section receiving source terms (the sum of the probabilities for each township equals one). Fields are placed randomly within the appropriate section at frequencies governed

by the section probability (but are still constrained by agriculturally capable land) and land that hasn't been used for any other source term.

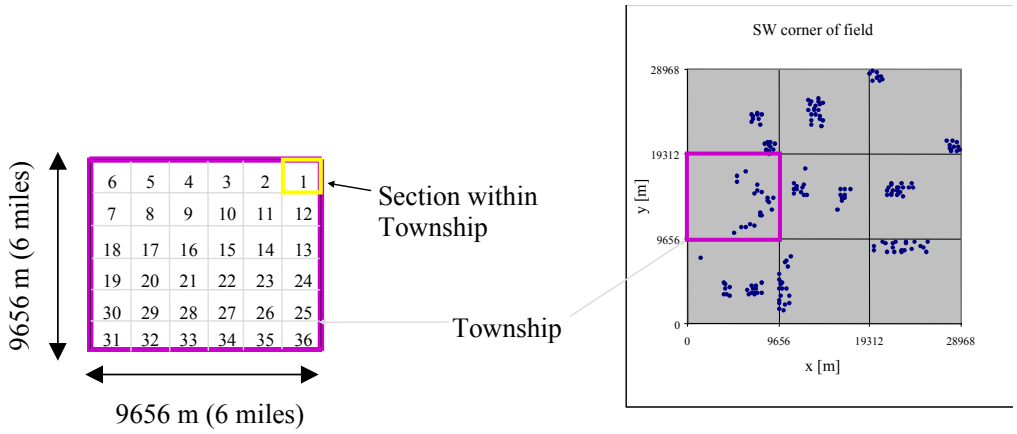

Figure 7. Township section numbering with example of section weighted field placement.

No other pesticide is rotated throughout the simulation cycle (i.e., all fields are always treated with same fumigant) for each consecutive year of the simulation.

Overflow of source terms to surrounding sections

It is possible that many township sections have historically received a large percentage of the allotted township allocation. If the current soil fumigant allocations were increased, and section weighting inputs are based upon historical records, it is conceivable that all the ag land within these sections can be treated before the fumigant mass allocation is achieved. This can occur if the section has a large percentage of non-ag capable land, the user has specified the township mass goes into relatively few sections, and/or when the total amount of pesticide mass (township allocation) is large. The latter constraint arises as more treated fields are required when the township allocation is increased. Thus, there is a need to account for high-use density sections where it is anticipated the highest air concentrations will occur. The southwest corner for a treated field (and thus area) is randomly selected for a section/township (Figure 7) via probability weighting assigned by the user for random placement. Once a field is placed, this area is unusable for additional applications of pesticide during the simulation year since a crop is now growing (currently, most fumigant applications are made pre plant). Thus, a "cookie cutter" scenario arises as different field sizes are randomly placed within a section (Figure 8). From a

numerical standpoint, there may be the possibility that a sufficiently large field cannot be placed in a given section, although the overall remaining area ag-capable land is of sufficient size. In these cases, a spill-over/overflow algorithm was developed. On subsequent iterations, if a small field size were selected, the algorithm would first try to place the field in the user-defined section before any overflow occurs to maximize the treated area in a user specified section.

Overflow Section Probabilities

The numerical model first attempts to place a specific field within a township section. A new random location is selected until the field can be placed or 100,000 iterations have occurred (subroutine "Get_Fld_Coordinates"). The overflow algorithm is initiated to place the field in a bordering section if at the end of these iterations a portion of unused land within a section was not found to place a field. A field may not be accommodated due to other field placement, or that it is close to the township boundary, or just will not geometrically fit in remaining ag-capable land. It is hypothesized that if a township section "fills up" such that no new treated fields can be placed, then the field should be placed in a section immediately surrounding this high density use area. Details of how the overflow algorithm works are as follows.

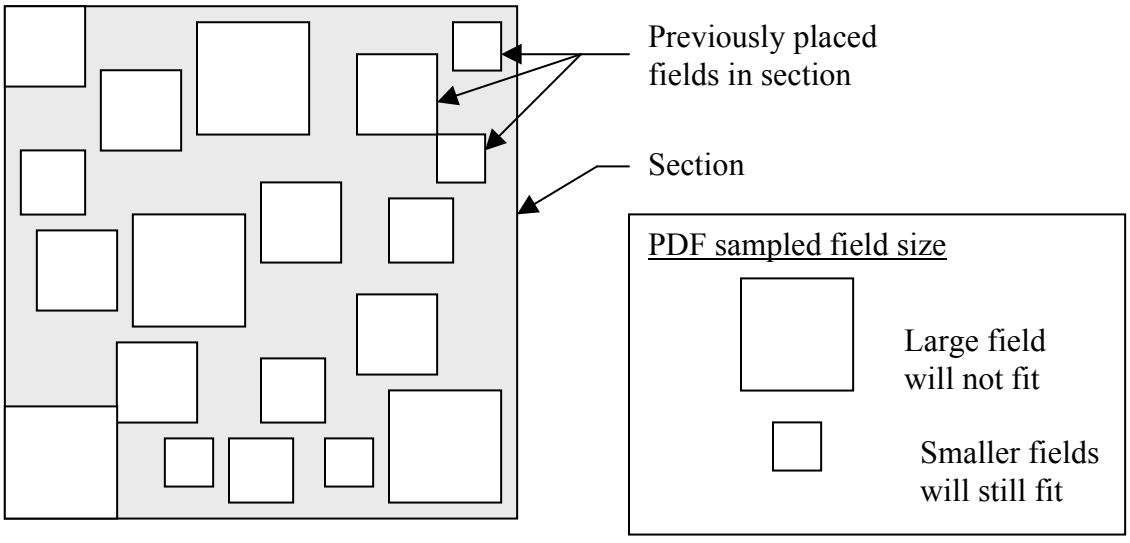

Figure 8. Illustration of field placement within a section/township where not all land can be utilized.

Before the onset of a simulation, the user has defined section probabilities for treated field assignment within a given township section. These section-weighting probabilities can be based upon historical records or expert judgment. For calculation purposes, each neighboring section surrounding the user specified non-zero probability section was internally assigned the same probability magnitude as the original section. The methodology is illustrated in the example found in Figure 9. The difference between Fig. 9 (a) and 9 (b) is that for 9 (a), all user defined section overflows don't impact (border) other potential overflow sections. In Figure 9 (a), the user has specified three township sections having non-zero probabilities of receiving a treated field (dark cells). All sections within the township that surround the user-supplied non-zero probability sections are initially assigned the same probability value (light cells). Then, each surrounding section probability is divided by the sum of all probabilities for the sections that border the user supplied non-zero probability sections. For the example provided, the numbers in parenthesis in the bordering sections is the actual probability assigned to the section.

All sections within the township that surround the user-supplied non-zero probability sections are initially assigned the same probability value (light cells). If a surrounding section borders multiple user supplied probability sections, then neighboring section probability is defined as the sum of the probabilities resulting from all neighboring sections. Then, each surrounding section probability is scaled by the sum of all probabilities for the sections that border the user supplied non-zero probability sections (Eq. 7).

$$P_i = \frac{NP_i}{\sum NP_i} \tag{7}$$

P_i = probability of a neighboring section receiving a treated field if the primary sections are filled up.

NP_i = Initial probability assigned to neighboring section surrounding a user-supplied non-zero probability section (same as bordering user supplied section).

ΣNP_i = sum of the initial probabilities for all neighboring sections within the township.

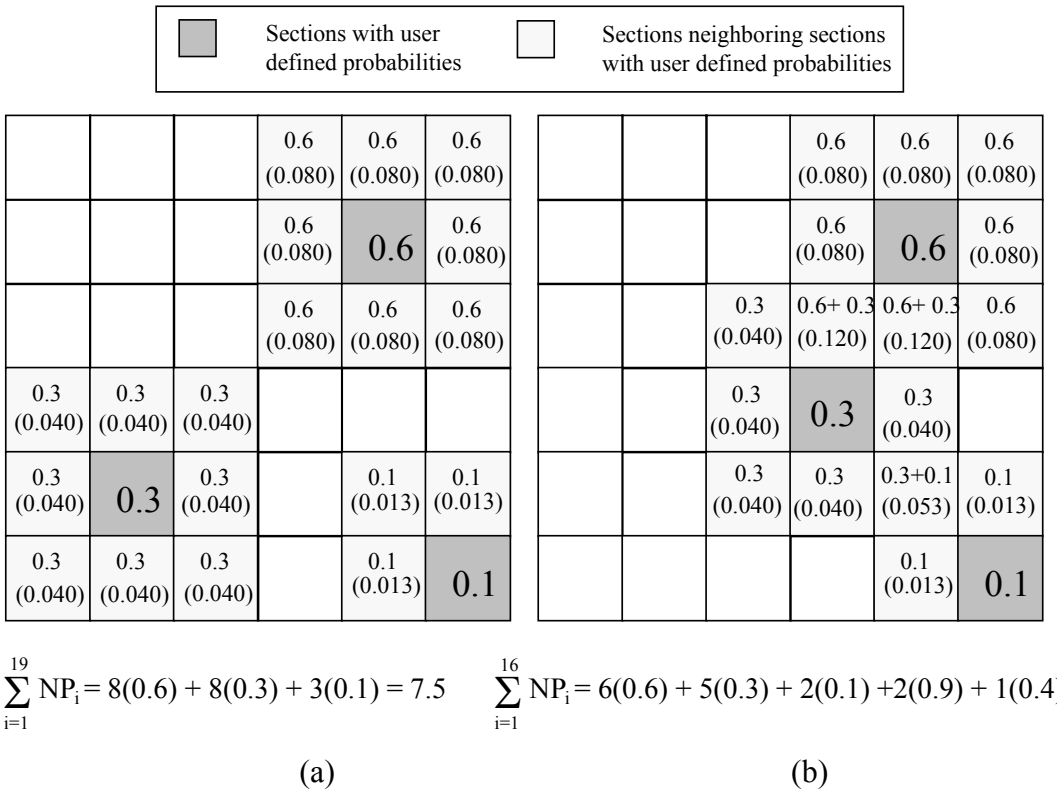

Figure 9. Example of the overflow algorithm for determining probability of surrounding sections that can receive a field if the algorithm is invoked. Initial magnitudes for surrounding sections (light cells) are given along with the resulting calculated probability (in parenthesis).

The methodology is slightly different when a neighboring section borders multiple user-defined sections [Fig. 9 (b)]. Here, the neighboring section probability is defined as the sum for all user-defined probabilities that the neighboring section borders. As before, these initial magnitudes (P_i) for neighboring sections are subsequently scaled by the sum of the probability magnitudes over all neighboring sections (NP_i) once each neighboring section has been assigned a defined magnitude (such that the sum of probabilities over all neighboring sections equals 1.0). In this way, neighboring sections near the largest magnitude user-defined section probabilities have the greatest chance of receiving an overflow field once the overflow algorithms are invoked. Similarly, neighboring sections that border multiple user defined sections will have a greater probability of receiving treated fields than the other neighboring sections that do not border multiple user defined sections. This approach appears consistent with affluent rural areas that are

capable of affording soil fumigation (i.e., a wealthy farmer who is already using soil fumigants and is increasing his/her land production or buying up neighboring land).

Figure 10 is an example of section weighting with overflow. User specified sections at the supplied probability weights are the first to fill with source terms. Once a section can no longer accommodate a field, then neighboring sections are used as the overflow area for field placement. If all overflow and surrounding sections fill, then source terms are randomly placed in any remaining ag-capable land within the township.

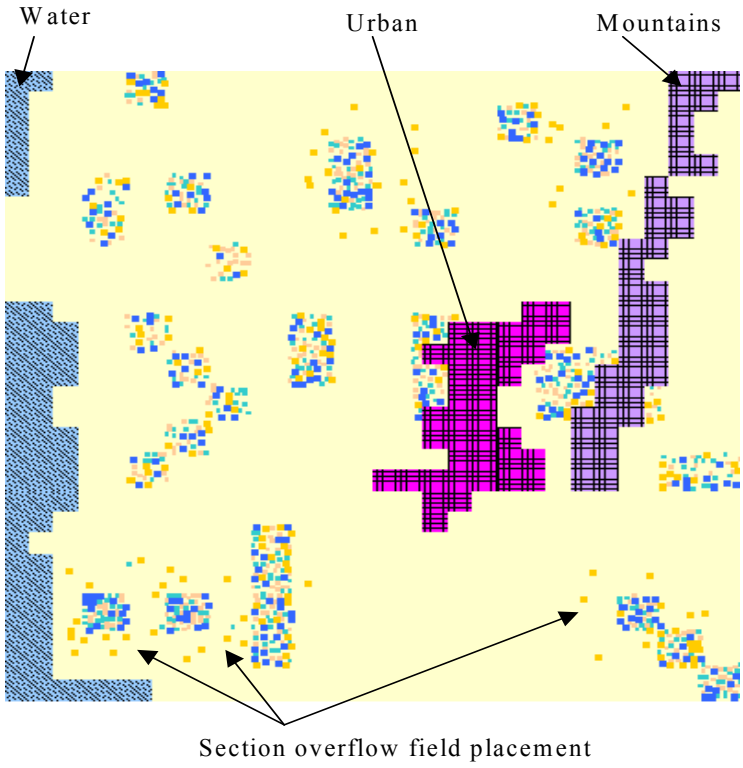

Figure 10. Example of user defined section weighting field placement with overflow allowed.

Section-Weighting Limitations

There are limitations to section weighting as currently implemented. The numerical model currently loops through crop type (up to 5 different crops) when placing fields. Thus, section weighting first places all fields for crop #1, followed by crop # 2, and so on. If the section fills up, the overflow algorithm will kick in and place fields in neighboring sections. It may be possible for the section area to fill up before all of the crop types have been addressed. This can

lead to artificial (or real) heterogeneity in crop types for a given region. The details of use data such as that found in the California Pesticide Use Records (PUR) data often do not distinguish crop type per section. Thus, the accuracy or error of this potential heterogeneous artifact for crop placement cannot be addressed at this time. However, field heterogeneity can be partially overcome through appropriate choice of section weighting as illustrated in Figure 11.

The graphics in Figure 11 represent a single township where the user has specified the probability of a township section receiving an application (upper portion of figure). The first graphic within this figure follows the user specification that all treated fields are placed in a single section. Since there is not enough land to encompass the amount of treated fields, the overflow algorithm is initiated and fields are placed in neighboring townships. The middle graphic gives much the same result as the first, but the user has specified a single section had the highest probability for treated fields, but neighboring townships also had a smaller probability. This center example is more homogenous in crop type then the first. Finally, the last graphic represents field placement results when non-continuous sections are given finite probabilities of receiving treated fields.

(a)

0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00

(b)

0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.20	0.00
0.00	0.00	0.00	0.00	0.60	0.00
0.00	0.00	0.00	0.00	0.20	0.00
0.00	0.00	0.00	0.00	0.00	0.00

(c)

0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.06	0.06	0.06
0.00	0.00	0.00	0.06	0.52	0.06
0.00	0.00	0.00	0.06	0.06	0.06
0.00	0.00	0.00	0.00	0.00	0.00

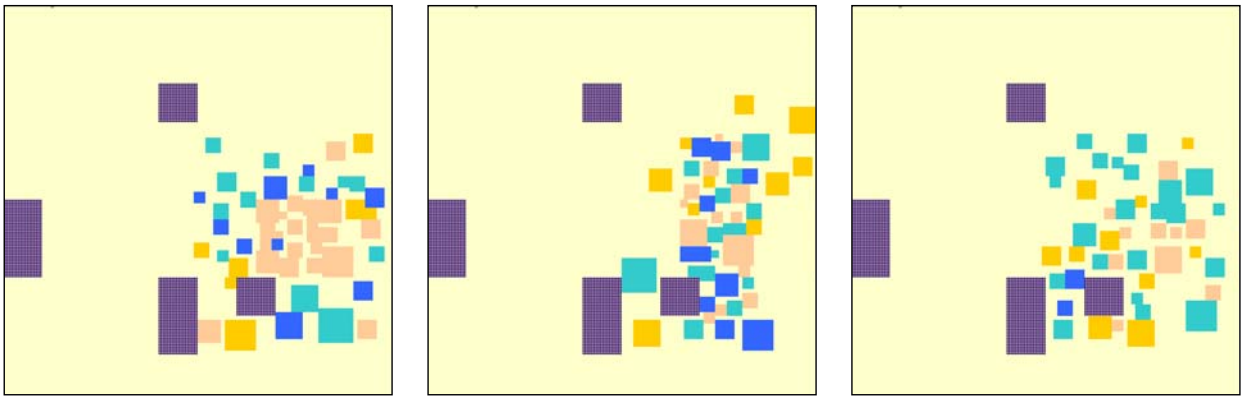

Figure 11. Example results from user specified section weighting with overflow.

Agricultural fields are typically fumigated once per year, with the exception of TV fields where applications occur once during the lifespan of the orchard or vineyard. A system was constructed such that any area that received a treatment during the year could not be retreated in keeping with agronomic practices. Also, any TV area was eliminated from further treatment for the rest of the multi-year simulation period. Land cover is based upon raster discretization of a township into a grid system 100x100 (10,000 equal sized cells). The southwest corner for the field is found by selection of a raster grid within the township. The grid selection is based upon section weighting probabilities or random placement in ag-capable lands. The field is then placed in the township, and surrounding grids are identified based upon the field location and size (assuming the land is ag-capable and no portion of the area has already been used for a prior source term).

Due to the discrete nature associated with a raster analysis, the number of surrounding grids a field occupies can be either "rounded" up or down, according to the user preference. Rounding up includes all grids a field comes in contact with, even if there is only slight overlap into a neighboring grid. Rounding down only includes a surrounding grid if more than half of the grid is occupied by the field. Figure 12 provides an example of the procedure in eliminating ag-capable land from further consideration in a simulation year if rounding up or down is assumed. Rounding down or up will give the same result if the field occupies at least 50% of a grid. By specifying rounding up, there will be no possibility for treated field overlap. A small probability of having some fields slightly overlap exists if rounding down is used. However, this may only occur for townships having a large township allocation and high field density in specific township sections. Rounding down can allow for denser field placement within a township section since more ag-land is available for further field placement.

The "Worst-Case" chronic exposure for an individual is dependent upon the time spent near a treated field immediately following a fumigant application which means that individuals residing near a field that is repeatedly treated with a soil fumigant year-in, year-out, will have the highest exposure potential. A critical input is therefore the density of fields within a township, the frequency of field retreatment from year to year, and the likelihood of individuals spending a large portion of their time near these treated fields during off-gassing.

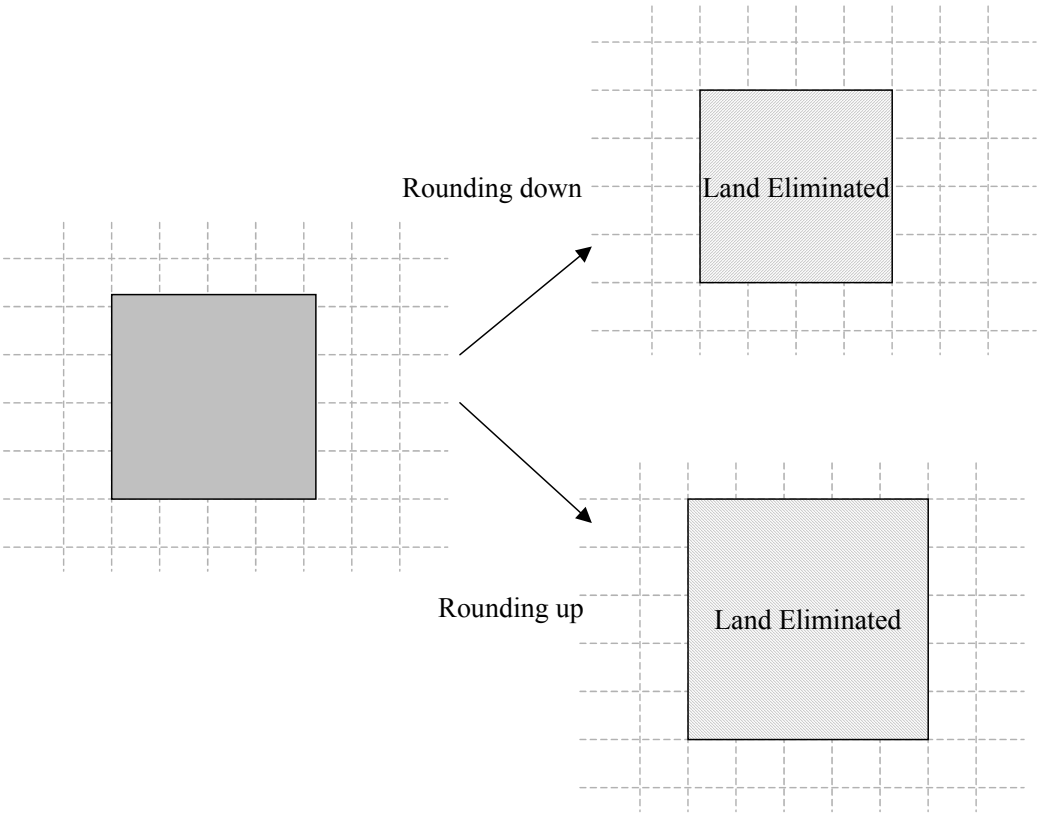

Figure 12. Example of land elimination due to integer rounding associated with field coverage.

The general flow diagram for the VBA code that is executed to place fields and generate appropriate source and flux files for ISCST3 is given in Figure 13. This diagram lists the main subroutines. There are other small subroutines that perform a small task that many of the subroutines seen in Figure 13 utilize. However, Figure 13 will direct programmers who may need to make modifications in the field placement/source strength subroutines.

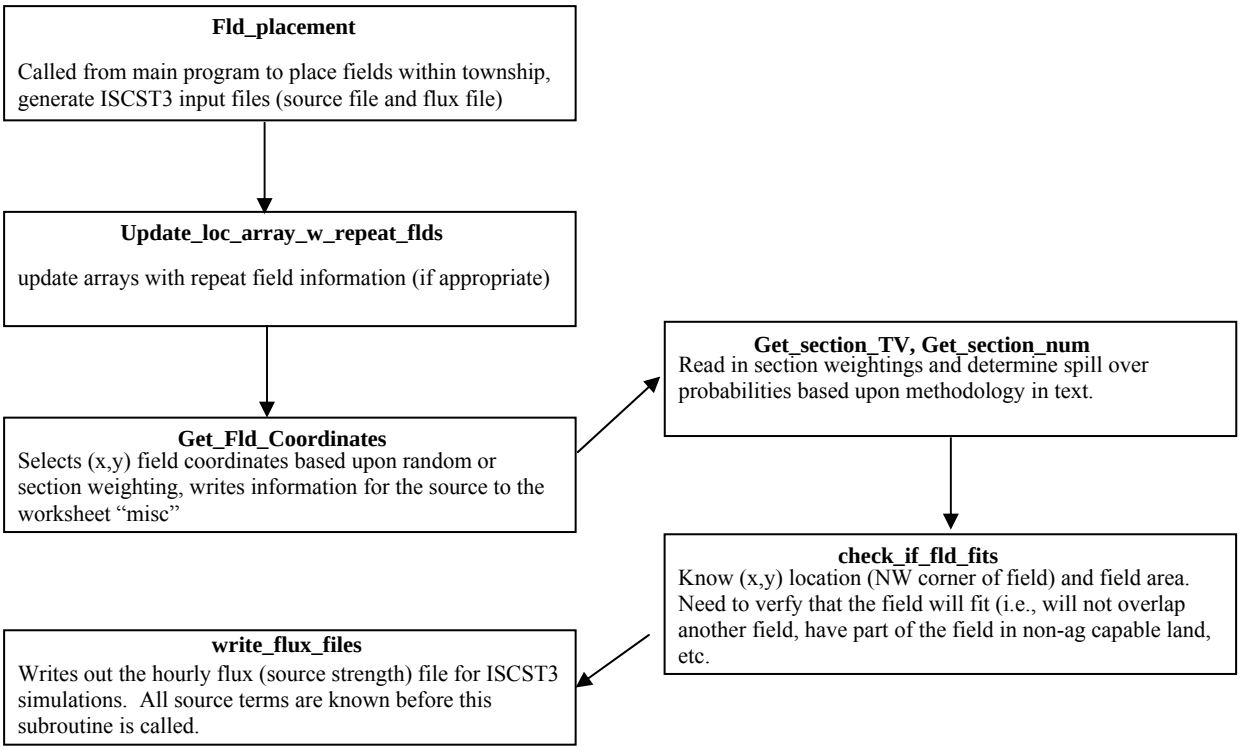

Figure 13. Flow diagram for field placement subroutines.

Field Retreatment in Subsequent Years

Figure 14 represents an example when 50% field retreatment is considered. For the first year of rotation, fields are placed appropriately (random or section weighted). Following the first year of simulation, 50% of the fields from the previous year are randomly selected (dark shade – Year 1) and marked as fields that will be retreated the following year (Dark shade – Year 2). New fields are now assigned (Light shade – Year 2) such that properties of retreated fields and new fields meet system constraints (see optimization section). This process is now repeated for each year of simulation.

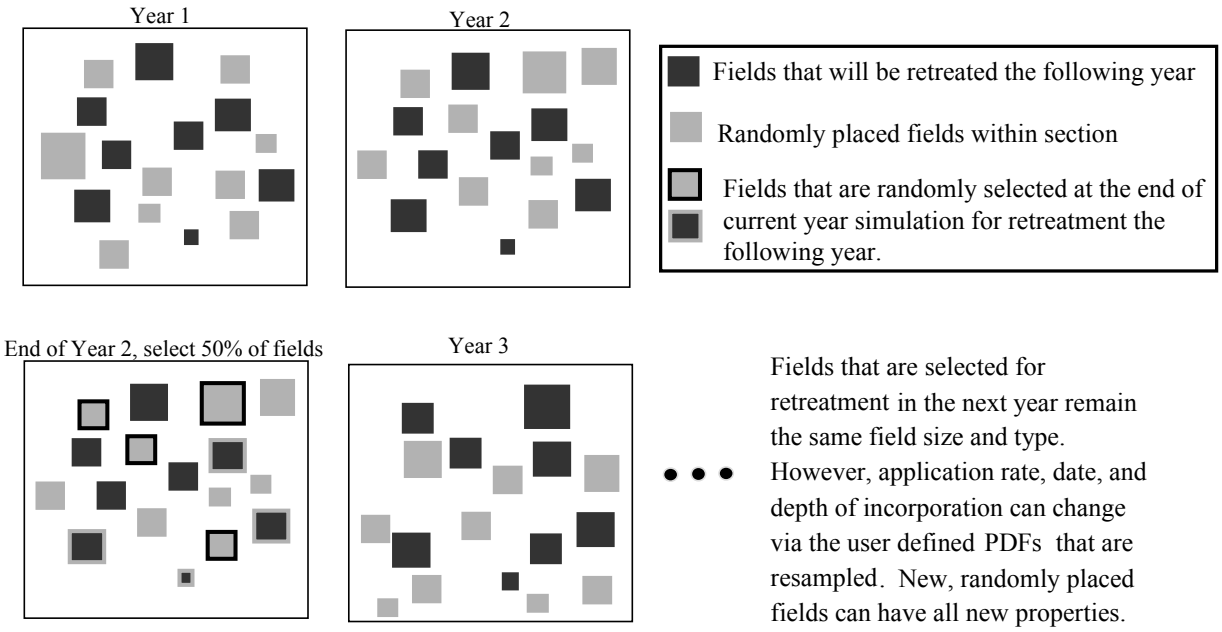

Figure 14. Example of field retreatment algorithm assuming 50% of fields from a given year will be retreated in the subsequent year.

In SOFEA, fields that will be retreated in the following year are flagged (stored in appropriate arrays) and are used in the next year of simulation to determine the township allocation. Thus, only the first year generates new fields, and subsequent years will use some fields from the previous year as well as new source locations for the current year. In addition, repeat field attributes are written to an ASCII file called "field_repeat". This file is used by the optimization program for the mass allocation contribution. Repeat fields are not "stretched or shrunk" in the optimization program. Only new fields for the simulation year are altered. In this way the exact same field size will be reused in next simulation year for field retreatment. The subroutine that randomly samples fields for retreatment for the next year of simulation is "Randomize_flds" and "fld_solve1".

Crop Type and Field Size Optimization

The stochastic nature of the numerical system dictates different field sizes and application rates will be selected for each crop type and for each year of simulation. A method was required to keep the fumigant mass applied to any given township (user specified) as a constant under any

condition. Thus, the township allocation is a management constraint that indirectly can control fumigant exposure.

Field Size Optimization

Optimization procedures were used such that the township allocation was achieved but constrained by the percentages of each crop type found within the township being met (crop percentages are also user supplied). User defined field size PDFs are initially sampled to obtain starting values for field sizes for each crop type. The number of fields are determined such that the total mass of soil fumigant for all source terms is constrained at the user specified township allocation and the residual for the crop percent cover is minimized. This results in a Mixed Integer Linear Program (MILP) problem whose optimal solution is obtained using a modified flexible-polygon search procedure (Himmelblau, 1972)

The mathematical representation for the objective function requiring minimization is given by Eq. 7.

$$\Psi = \gamma \sum_{i=1}^{5} \left| T_{pdf_i} - \frac{\sum_{j=1}^{N_i} A_{i,j}}{\sum_{i=1}^{5} \sum_{j=1}^{N_i} A_{i,j}} * 100 \right| - \left| T_{cap\text{-}adj} - \sum_{i=1}^{5} \sum_{j=1}^{N_i} A_{F\,i,j}\, A_{i,j}\, R_{i,j} \right| \tag{7}$$

Ψ = Objective function requiring minimization

γ = Weighting variable (100 or 1000) based upon order of magnitude analysis so optimization procedure executes properly under a variety of diverse conditions. By adjusting γ, one can emphasize the crop percent residual, township allocation residual, or both)

$T_{pdf\,i}$= Percent of township ag-capable land that is specifically for crop "i" (from PDF)

T_{cap}= Township allocation for soil fumigant [kg]

N_i = Integer number of fields for crop "i" (initially unknown)

A_i = Area of a field for crop "i" [ha]

R_i = Application rate for crop "i" [kg ha^{-1}]

i = counter for the 5 different crop types that can be present

The first and second term in Eq. 2 represents the sum of residuals for cropping area percentages and for the township allocation, respectively. Thus, Eq. 2 is a function of the township allocation and the percentages of user defined field types within the township. The numbers of fields for each crop type (N_i) are constrained as integer's ≥ 0. Once the number of fields for a given crop type are known, then the PDF field sizes (A_i) are adjusted (slightly stretched or shrunk) to meet the township allocation constraint. The logic flow diagram used in optimization procedure is given in Figure 15. The parameter γ is used to emphasize one constraint over the other (or to make each constraint of similar magnitude during optimization). Unknown values before the optimization begins are the number of fields for each crop type (Ni), and the ending field size for each field "j" for crop "i" ($A_{i,j}$). A total of 6 calls to the optimization routine are used to provide initial estimates for the decision variables, followed by refinement of these variables. For the first four calls, the weight factor in the objective function (γ) was 1000, 100, 10 and 1 respectively, to first provide a crude estimate for decision variables and then to consecutively refine this initial estimate. Since γ begins as a large value, the percent of overall crop types dominates the objective function, and the optimization routine focuses on estimating N_i. A value of $\gamma = 1000$ is used to supply the "initial" guess to N_i. As γ decreases from 1000 to 1, the number of fields (N_i) is altered as the township allocation portion of the objective function becomes of similar importance. When $\gamma = 1$, and both the residuals associated with user specified crop percentages and the township allocation are minimized. The procedure for first estimating N_i followed by refinement in this estimate is outlined in the upper loop of Fig. 15. For $\gamma < 1$ (lower loop of Fig. 21), changes in the number of fields (N_i) will remain small as the optimization procedure now "stretches" or "shrinks" the field sizes such that the township allocation constraint is achieved. Field size lower and upper bounds ($A_{i,j\ lower}$, $A_{i,j\ upper}$) for optimization were constrained at ± 20% of the PDF sampled size.

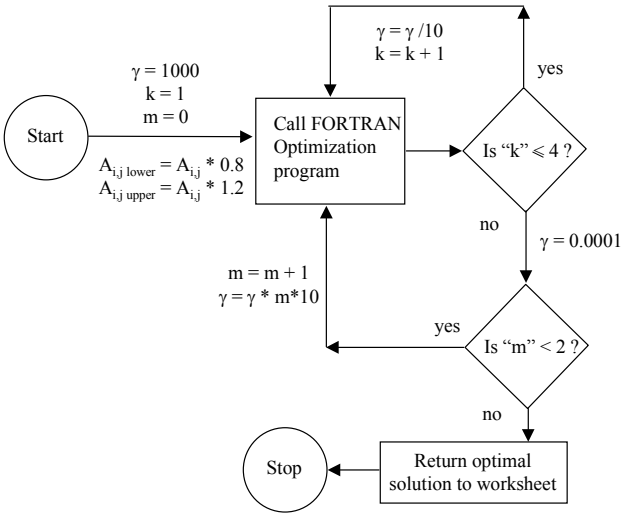

Figure 15. Flow diagram for logic used in optimization procedure for determining both the number of fields for a given crop "i' (N_i) and how these fields need to be stretched or shrank to obtain the user specified adjusted mass constraint for a given township.

The optimization program source code (Written in Fortran) is given in Appendix A. Opt2.exe is called for the central 3x3 townships, and for each township where fumigant mass has been allotted outside the central 3x3. Input files required by the optimization program ("opt2.exe") are written from VBA, the program opt2 is executed, and results are imported back into VBA (all transparent to the user). Input files required by opt2.exe are "field.dat" and field_repeat.dat" and are written to the bin directory. The ASCII output file generated by opt2.exe and subsequently read back into VBA is called "field_opt_dat". Figure 16 is the FORTRAN Code reading in the ASCII file "field.dat". Parameters of this file are defined as follows:

The fields "field.dat", "field_repeat.dat" and "field_opt.dat" are all free formatted.

```
      READ(1,*) TWN_CAP, ITERATION, K_READ, TWN_1_9
*
* TWN_1_9 IS THE INTEGER FLAG TELLING THIS PROGRAM HOW MANY TOWNSHIPS TO FIND
* SOURCE FILES FOR.  TWN_1_9 EQUALS EITHER 1 OR 9.  IF = 9, THEN WE ARE FINDING
* SOURCE TERMS FOR THE CENTRAL 3X3 TOWNSHIP DOMAIN.  IF = 1, THEN WE ARE FINDING
* SOURCE TERMS FOR A SINGLE TOWNSHIP THAT IS "OUTSIDE" OF THE CENTRAL 3X3.
*
* Note: if K_READ =  0, then we need to read in information from the file
* "field_repeat.dat" that contains information about the fields that will
* need to be retreated for this year of simulation.

* READ IN THE FRACTION OF THE REFERENCED TOWNSHIP ALLOCATION FOR EACH TOWNSHIP ID
      READ(1,*) (FRAC_CAP(I),I=1,TWN_1_9)
```

```
* READ IN THE CROP PERCENTAGE AREAS FOR EACH TOWNSHIP
      DO J = 1, TWN_1_9
        READ(1,*,END=999)(CROP_PER(J,I),I=1,5)
      END DO

* READ IN THE CRYSTAL BALL SAMPLING FOR AREA, APPLICATION RATE, DEPTH OF INCORP.
* ACUTAL MASS APPLIED, CORRECTED MASS (BASED ON DEPTH AND TIME OF YEAR)
      DO I = 1,5000
        READ(1,*,END=999,ERR=998)( AREA(J,I), RATE(J,I), DEPTH(J,I),
     #        IDATE(J,I),AMASS(J,I), CMASS(J,I), D_or_S(J,I),
     #        ITARP(J,I), SINCORP(J,I),SYR(J,I),J=1,5)
      END DO
```

Figure 16. FORTRAN Code from opt2.f where the program reads in the ASCII file "field.dat".

Parameters in Figure 16 are:

FRAC_CAP(I),I=1,TWN_1_9) = fraction of the referenced township allocation for each township id
CROP_PER(J,I),I=1,5) = crop percentage areas for each township, j=1 to TWN_1_9
AREA(J,I) = field area (ha)
RATE(J,I) = application rate (kg/ha)
DEPTH(J,I) = depth of incorporation (cm)
IDATE(J,I) = Julian application date
AMASS(J,I) = Actual mass [kg]
CMASS(J,I) = corrected mass (based upon CDPR methodology) [kg]
D_or_S(J,I)= Flag for drip (<2) or shank (=2) application
ITARP(J,I)= Flag if a tarp is (<2) or is not (=2) present
SINCORP(J,I)= Scaling factor based upon soil incorporation depth
SYR(J,I) = Scaling factor based upon time of year (CDPR methodology)

Field_repeat.dat

The following line is from opt2.exe that read in the ASCII file "field_repeat.dat".

```
READ(3,*,END=1999,ERR=1998)TWN_ID, CRP_ID, XR, YR, ZR, AR, RR,
DR, IR, AM, CM, DOS, ITR, SINCR, SYRR
```

Twn_ID = township id (either 1-9, or some number from 1 – 529 for external (to 3x3) townships
Crp_ID = integer crop ID (1-5, where 1 = TV, 2 = FC, etc.)
(XR, YR,ZR) = (x,y,z) location of Northwest corner of field
AR = field size
RR = application rate
DR = application depth

IR = application date (Julian)
AM = actual mass
CM = corrected mass
DOS = Drip_or_Shank flag
ITR = tarp flag (ITARP)

Field_opt.dat

'The format for "field_opt.dat" is as follows. There is a "grouping of information for each township. The first line of data for a group contains the
Township ID #TV #FC #NC #SB # PP , where #TV = total number of tree and vine fields, etc.

The next (#TV + #FC + #NC + #SB + #PP) lines of information contain data about the fields. The first #TV lines are for Tree and vine fields, the next #FC lines are for Field crop fields, etc. Information on these lines are as follows:

Township ID Area_start Area_end app. rate Scal. factor Depth-incorp app. Date Drip/Shank Tarp(Y/N) Sincorp Syr Fld repeat x y z

Area_start = starting field size before optimization (i.e., that sampled from the appropriate PDF
Area_end = ending field size after optimization
Scal. factor = CDPR factor for time of year and application depth (This is the # used for correcting fumigant mass
Depth-incorP = depth of incorporation
app. Date = Julian application date
Drip/Shank = if application is drip (0) or shank (1)
Tarp = if bare field (0) or tarp is present (1)
Sincorp = flux scaling factor for incorporation depth
yr = flux scaling factor for time of year
Fld_repeat = Flag for if this field is to be repeated in the next year of simulation (0 = no, 1 = yes)
x, y, z = NW corner field coordinates for repeat fields (if Fld_repeat = 0, then these entries are blank)

Temporal parameter changes

Agronomic practices can change over time. These practices can include such things as the percentage of retreated fields from year to year, field size changes, the amount and type of pesticide used, application rates, incorporation depths, and loss of ag-capable land as cities grow.. In addition, the ability to explore heuristic rules that may mitigate fumigant exposure is desirable (such as a field cannot be treated 3-years in a row, staggering of sources between

township sections in alternating years, and so on). Thus, a system with the ability to forecast an exposure regime was required that could account for not only current scenarios but future "what-if" scenarios as agronomic practices may change.

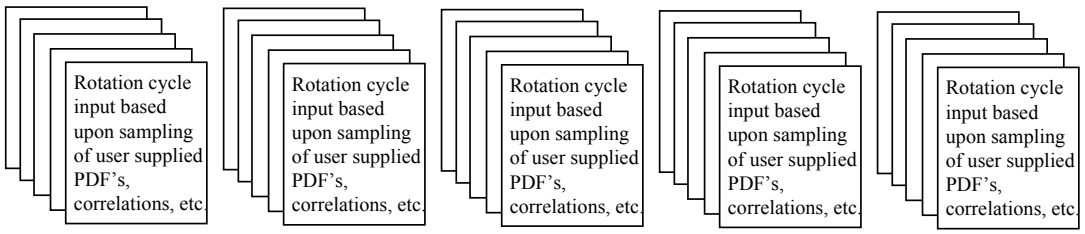

Figure 17. Example of a 5 period rotation cycle, where each period has 5 years of simulation (25-yr simulation).

Figure 17 represents how the temporal nature of agriculture can now be simulated. A 25-year simulation has been broken down into five different 5-year intervals. The first 5 years represents current conditions (i.e., actual use data that has been collected). The next five years are "near term" approximations to agronomic practices, where input parameters are altered slightly (such as increased use of a particular fumignat as a replacement for MeBr). The following 15-years incorporate land use/demographics that may occur such as urbanization, ending of the life span of current orchards (i.e., new areas for TV), potentially new BMPs regarding insect resistance management, an so on. The same 25-year simulation can be equally broken up into three distinct intervals (where parameters are statistically different than in other intervals) of 8 years, 8, years, 9 years, and so on, to address parameter assumptions and their impact on forecasting results.

The forcast“ looping’ is generated simply by a “do” loop in VBA. The maximum number of loops that can be incorporated is five. This is due to the hard coding for input parameter sampling from the various worksheets of SOFEA.

Superposition of Unique Crop type results

Summarized ISCST3 output includes 24-hr maximum, and annual receptor concentrations. Post processing routines were written such that additional averaging periods can be specified by the user (i.e., 3-day, 15-day, and so forth). Air concentrations are obtained for each crop type independently. Once all emission losses/air concentrations for each crop type are simulated, air concentrations are superimposed to obtain representative air concentrations just as if all crop types were simulated together (Figure 18).

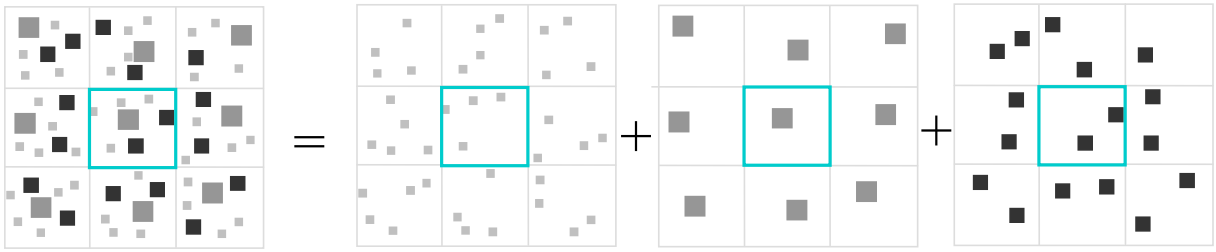

Figure 18. Superposition of daily simulation results for each crop type for the simulation interval (1-year) for an example having 3-crop types.

The reasons for superposition of simulation results are as follows. The execution time for ISCST3 scales linearly with the number of source terms and the number of receptors. When current township allocations are approached or exceeded, and the field sizes are small, then a relatively large number of sources for the simulation domain will exist. Likewise, the total number of receptors placed within the central 3x3 (9-township) simulation domain can become large as the resolution/spacing between neighboring receptors becomes small. Additionally, MC sampling requires an appropriate number of yearly simulations for stochastic response surface generation since parametric uncertainty is characterized through MC simulations. Thus, a simulation was broken up into yearly events for each field type to keep I/O and simulation execution time manageable. Simulation results for each crop type were superimposed (i.e., added) on a daily basis for each receptor within the simulation domain. Results can be

superimposed since Gaussian dispersion is independent of concentration gradients (i.e., the wind convects the pesticide mass with dispersion as specified by the directional dispersion coefficients). Simulation results are stored in memory (arrays) and superimposed in subroutines "Post_Process", "print_chronic", and "print_simulation_avg".

Post Processing of ISCST3 Simulation Results

Simulation results (air concentrations) generated and written to ISCST3 output files are read in via VBA and are stored in memory upon completion of a simulation year for a given crop. After each simulation, 24-hr maximum and chronic (annual) air concentration results are stored in memory and written to the worksheets "24hr_max" and "Chronic", respectively. Once all crops have been simulated, a call to the subroutine "Post_Process" is made where the running averages specified by the user are determined and written to the worksheet "Run_avg_twn" and "24hr_summary". Results in "24hr_summary" are for each crop type, while those found in "Run_avg_twn" are the superimposed results from all crop types. Results found in "Run_avg_twn" can be generated by the individual crop results found int "24hr_summary".

Once all of the user specified number of years for the simulation has completed, results stored in memory are then used to generate an N-year average value for chronic and subchronic exposure intervals via calls to the subroutines "print_simulation_avg" and "print_chronic ". Results from these two subroutines are found as the last columns of numbers in the worksheets "Chronic" and "Run_avg_twn". Subroutine definitions are found in Appendix B. Subroutine connectivity and parameters used in SOFEA are summarized in Appendices C and D, respectively.

Conclusions

SOFEA is a comprehensive numerical tool for simulating air exposure concentrations resulting from fumigant use. SOFEA is written in VBA, although several FORTRAN programs are used (OPT2 and ISCST3). VBA is effectively used with MS Excel to provide a pre- and post-processor capability for ISCST3 such that the details of the repetitive tasks are transparent to the user. This manuscript serves as a resource for programmers who may require modifications be made to SOFEA, in addition to the functionality and the algorithms used throughout.

References

Cryer, S.A., I.J. van Wesenbeeck, and J.A. Knuteson, 2003. Predicting Regional Emissions and Near-Field Air Concentrations of Soil Fumigants Using Modest Numerical Algorithms: A Case Study Using 1,3-Dichloropropene. J. Agric. Food Chem, 51, 3401-3409.

Cryer, S.A. and I. van Weseenbeck. 2001. Predicted 1,3-Dichloropropene Air Concentrations Resulting from Tree and Vine Applications in California. J. Environ. Qual. 30:1887-1895.

Cryer, S.A., 2004 (In Preparation). Predicting Soil Fumigant Acute, Sub-chronic, and Chronic Air Concentrations under Diverse Agronomic Practices.

ISCST3. 1995. Users Guide for the Industrial Source Complex (ISC3) Dispersion Models, Vol. I – Users Instructions. U.S. Environmental Protection Agency, Office of Air Quality Planning and Standards Emissions, Monitoring, and Analysis Division, Research Triangle Park, North Carolina 27711, EPA-454/B-95-003a, Sept. 1995.

van Wesenbeeck, I.J. , S. A. Cryer, B.A. Houtman, and P.L. Havens 2004 (In Preparation). Managing 1,3-Dichloropropene Soil Fumigant Exposure/Risk in California

Appendices

Appendix A. FORTRAN Source Code for Optimization Program

```
************************************************************************
*                                                                      *
*                               opt2.f                                 *
*                                                                      *
*  This optimization routine determines the number of fields for each crop    *
*  type for the fumignat management system.  The objective function is to     *
*  minimize the residual between the user specified crop area for each crop,  *
*  and the user specified township allocation.  The township allocation in this case is the *
*  corrected cap based upon time of year an application is made and the depth *
*  of incorporation.  The corrected mass is calculated in the VBA code of the *
*  MS Excel workbook, and VBA creates the file "township.dat" for every year  *
*  of simulation.  Information in this file includes 1) Corrected township allocation*
*  basis, the fraction of the township allocation basis for each of the 9 townships  *
*  to be simulated, and up to 750 randomly sampled field information for      *
*  each crop type.                                                         *
*                                                                      *
*  We now read in information for fields  that are to be reused from a        *
*  subsequent year of simulation and still optimize # of fields and sizes     *
*  such that the corrected township allocation is acheived and the user specified    *
*  percentages for each crop type are approached.  Both "new" and "repeat"    *
*  fields are written to the file "field_opt.dat"                          *
*                                                                      *
*  Written by Steve Cryer                                              *
*  Version 1.0                                              10/30/02    *
*                                                                      *
************************************************************************
*
      PARAMETER(MAX=800, MAX1 = 80)
      IMPLICIT REAL*8(A-H, O-Z)
      REAL*8 X(20), XL(20), XU(20), FUN(5), X_AREA(MAX),
     1       XL_AREA(MAX), XU_AREA(MAX), X_AREA_START(MAX),
     2       X_DEPTH(MAX),SINCORP(5,MAX),SYR(5,MAX),
     3       SF_INCRP(MAX), SF_YR(MAX),SINCORP_R(9,5,MAX1),
     4       SYR_R(9,5,MAX1)
*
      INTEGER IDATE, IX_NEW1(20), IX_OLD1(20), DS(MAX), TARP(MAX),
     1               JULDATE(MAX),IMAX(5), D_or_S(5,MAX),
     2               D_or_S_R(9,5,MAX1), TWN_ID, TID, CID,
     3               JX(9,5), ITOT(5), CPR_I, KX(5), ITOT_FLDS(9),
     4               FLD_ID, FR, K_READ,TWN_1_9

      COMMON /FITCOM/ EPSP, EPSF, STEP, ALPHA, BETA, GAMMA

      COMMON /MISC/ CROP_PER(9,5),AREA(5,MAX),RATE(5,MAX),
     #       DEPTH(5,MAX),IDATE(5,MAX),AMASS(5,MAX),CMASS(5,MAX),
     #       LCOUNT,TWN_CAP, FRAC_CAP(9),WT_FACT,IX_OLD(20),
     #       RESID1,RESID2,IX_NEW(20),ITARP(5,MAX),ITERATION

      COMMON /MISC1/ X_RATE(MAX), X_AF(MAX), IX(20),RESID3, RESID4

      COMMON /MISC2/ AREA_R(9,5,MAX1), RATE_R(9,5,MAX1),
     1 DEPTH_R(9,5,MAX1), IDATE_R(9,5,MAX1), AMASS_R(9,5,MAX1),
     2 CMASS_R(9,5,MAX1), ITARP_R(9,5,MAX1), X_R(9,5,MAX1),
     3 Y_R(9,5,MAX1), Z_R(9,5,MAX1), JX, AF_R(9,5,MAX1)


      OPEN(1,FILE="field.dat",STATUS="OLD",FORM="FORMATTED")
      OPEN(2,FILE="field_opt.dat",STATUS="UNKNOWN",FORM="FORMATTED")
*
* READ IN THE TOWNSHIP ALLOCATION BASIS (KG), and iteration year of simulation.
* If iteration > 1, then we will have repeat fields that read in from
```

```
* the file "field_repeat.dat"
*
      READ(1,*) TWN_CAP, ITERATION, K_READ, TWN_1_9
*
* TWN_1_9 IS THE INTEGER FLAG TELLING THIS PROGRAM HOW MANY TOWNSHIPS TO FIND
* SOURCE FILES FOR.  TWN_1_9 EQUALS EITHER 1 OR 9.  IF = 9, THEN WE ARE FINDING
* SOURCE TERMS FOR THE CENTRAL 3X3 TOWNSHIP DOMAIN.  IF = 1, THEN WE ARE FINDING
* SOURCE TERMS FOR A SINGLE TOWNSHIP THAT IS "OUTSIDE" OF THE CENTRAL 3X3.
*
* Note: if K_READ =  0, then we need to read in information from the file
* "field_repeat.dat" that contains information about the fields that will
* need to be retreated for this year of simulation.

* READ IN THE FRACTION OF THE REFERENCED TOWNSHIP ALLOCATION FOR EACH TOWNSHIP ID
      READ(1,*) (FRAC_CAP(I),I=1,TWN_1_9)

* READ IN THE CROP PERCENTAGE AREAS FOR EACH TOWNSHIP
      DO J = 1, TWN_1_9
        READ(1,*,END=999)(CROP_PER(J,I),I=1,5)
      END DO

* READ IN THE CRYSTAL BALL SAMPLING FOR AREA, APPLICATION RATE, DEPTH OF INCORP.
* ACUTAL MASS APPLIED, CORRECTED MASS (BASED ON DEPTH AND TIME OF YEAR)
      DO I = 1,5000
        READ(1,*,END=999,ERR=998)( AREA(J,I), RATE(J,I), DEPTH(J,I),
     #        IDATE(J,I),AMASS(J,I), CMASS(J,I), D_or_S(J,I),
     #        ITARP(J,I), SINCORP(J,I),SYR(J,I),J=1,5)
      END DO

998   PRINT *, 'ERROR IN LINE ', I
      PAUSE

999   NTOT = I - 1
      CLOSE(1)

* READ IN THE FILE CONTAIING INFORMATION ON REPEAT FIELDS FOR CURRENT YEAR
* OF SIMULATION
*``````````````````````````````
*      IF (ITERATION .GT. 1) THEN
      IF (K_READ .EQ.  0) THEN
*``````````````````````````````
      DO J = 1, TWN_1_9
        DO KK =1, 5
          JX(J,KK) = 0
        ENDDO
      ENDDO

      TID = 1
      CID = 1
      KCOUNT = 1

      OPEN(3,FILE="field_repeat.dat",STATUS="OLD",FORM="FORMATTED")
*     =============
      DO I = 1,5000
*     =============


      READ(3,*,END=1999,ERR=1998)TWN_ID, CRP_ID, XR, YR, ZR,
     1         AR, RR, DR, IR, AM, CM, DOS, ITR, SINCR, SYRR

* DETERMINE THE NUMBER OF RETREATED FIELDS FOR EACH CROP TYPE AND EACH
* TOWNSHIP ID.  THIS INFORMATION (# OF RETREATED FIELDS) IS STORED IN THE
* ARRAY "JX(TWN ID, CROP ID)", WHERE TWN_ID = 1 TO 9, AND CROP_ID = 1,5

      IF (TWN_ID .EQ. TID) THEN

           IF (CRP_ID .EQ. CID) THEN
             JX(TWN_ID,CRP_ID) = KCOUNT
           ELSE
             CID = CRP_ID
             KCOUNT = 1
```

```
            JX(TWN_ID,CRP_ID) = KCOUNT
          ENDIF

         X_R(TWN_ID,CRP_ID,KCOUNT)= XR
         Y_R(TWN_ID, CRP_ID,KCOUNT)=YR
         Z_R(TWN_ID, CRP_ID,KCOUNT)=ZR
         AREA_R(TWN_ID,CRP_ID,KCOUNT)=AR
         RATE_R(TWN_ID, CRP_ID,KCOUNT)=RR
         DEPTH_R(TWN_ID, CRP_ID,KCOUNT)=DR
         IDATE_R(TWN_ID, CRP_ID,KCOUNT)=IR
         AMASS_R(TWN_ID, CRP_ID,KCOUNT)=AM
         CMASS_R(TWN_ID, CRP_ID,KCOUNT)=CM
         D_or_S_R(TWN_ID, CRP_ID,KCOUNT)=DOS
         ITARP_R(TWN_ID, CRP_ID,KCOUNT)=ITR
         SINCORP_R(TWN_ID, CRP_ID,KCOUNT)=SINCR
         SYR_R(TWN_ID, CRP_ID,KCOUNT)=SYRR
         AF_R(TWN_ID, CRP_ID,KCOUNT)=CM/AM
         KCOUNT = KCOUNT + 1

       ELSE

       TID = TWN_ID
       KCOUNT = 1
            IF (CRP_ID .EQ. CID) THEN
              JX(TWN_ID,CRP_ID) = KCOUNT

            ELSE
              CID = CRP_ID
              KCOUNT = 1
              JX(TWN_ID,CRP_ID) = KCOUNT

            ENDIF

         X_R(TWN_ID,CRP_ID,KCOUNT)= XR
         Y_R(TWN_ID, CRP_ID,KCOUNT)=YR
         Z_R(TWN_ID, CRP_ID,KCOUNT)=ZR
         AREA_R(TWN_ID,CRP_ID,KCOUNT)=AR
         RATE_R(TWN_ID, CRP_ID,KCOUNT)=RR
         DEPTH_R(TWN_ID, CRP_ID,KCOUNT)=DR
         IDATE_R(TWN_ID, CRP_ID,KCOUNT)=IR
         AMASS_R(TWN_ID, CRP_ID,KCOUNT)=AM
         CMASS_R(TWN_ID, CRP_ID,KCOUNT)=CM
         D_or_S_R(TWN_ID, CRP_ID,KCOUNT)=DOS
         ITARP_R(TWN_ID, CRP_ID,KCOUNT)=ITR
         SINCORP_R(TWN_ID, CRP_ID,KCOUNT)=SINCR
         SYR_R(TWN_ID, CRP_ID,KCOUNT)=SYRR
         AF_R(TWN_ID, CRP_ID,KCOUNT)=CM/AM
         KCOUNT = KCOUNT + 1

       ENDIF


*      =============
       END DO
*      =============


1998   PRINT *, 'ERROR IN LINE ', I , ' "FIELD_REPEAT.DAT" '
       PAUSE

1999     NTOT = I - 1

*        PRINT *, AREA_R(9,5,1) *** JUST A CHECK TO MAKE SURE IT READS FILE O.K.
*        PAUSE
         CLOSE(3)
*
*``````````````````````````````
       ENDIF
*``````````````````````````````
* PRINT OUT NUMBER OF REPEAT FIELDS FOR EACH TOWNSHIP AND EACH CROP TYPE
* NOTE: THIS WAS USED FOR TROUBLE SHOOTING AND SHOULD BE COMMENTED OUT
```

```
* FOR NORMAL EXECUTION.
*
*      DO J = 1, TWN_1_9
*        DO KK = 1, 5
*          print *, 'JX(',J,',',KK,') = ',JX(J,KK)
*        ENDDO
*      PAUSE
*      ENDDO
*
* Estimate lower and upper bounds on parameters.  For our case, the upper
* and lower bounds represent the bin interval range for droplet radius.
*
      XL(1) = 0.
      XL(2) = 0.
      XL(3) = 0.
      XL(4) = 0.
      XL(5) = 0.
*
      XU(1) = 50.0
      XU(2) = 50.0
      XU(3) = 50.0
      XU(4) = 50.0
      XU(5) = 50.0
*
* Set the convergence criteria (i.e., if LOOP = 1.  IF loop = 0, program
* uses defaults.

      LOOP = 1

      IF(LOOP .EQ. 1) THEN
       EPSP=1.0E-8
        EPSF=1.0E-0
         STEP=1.0
         ALPHA=1.0
         BETA=0.5
       GAMMA=2.0
      ENDIF

      NX = 5
      NY = 0
*

      DO I = 1, 5
        IX_OLD(I) = 1
        IX_OLD1(I) = 1
        IMAX(I) = 0    !IMAX = number of fields for crop type "I"
                       !       for all townships simulated
      ENDDO



      PRINT *,'               TV  FC  NC  SB  PP   CROP_RESID CAP_RESID'
*
*======================#########
      DO LCOUNT = 1, TWN_1_9          ! This is the counter for Township ID
*======================#########
      KTOT = 0
*

      DO KK = 1, 5
        X(KK) = 1
        IF(CROP_PER(LCOUNT,KK) .LE. 11.0) X(KK) = 0.0
      ENDDO

*-------------------------------------------------
*NO NEED TO RUN OPTIMIZATION ROUTINE IF THE MASS
* APPLIED TO THE TOWNSHIP IS ZERO
      IF (TWN_CAP * FRAC_CAP(LCOUNT) .LE. 0.0001)Then
        DO KK = 1, 5
         X(KK) = 0
        ENDDO
```

```
      GOTO 1001
      ENDIF
*-----------------------------------------------

      LOOP = 1
*     ''''''''''''''''''''
      IF(LOOP .EQ. 1) THEN
       EPSP=1.0E-01
        EPSF=1.0E-03

         STEP=1.1

         ALPHA=1.1
         BETA=0.5
       GAMMA=0.00001
      ENDIF
*     ''''''''''''''''''''

      LOOP = 1
      WT_FACT = 1000.0

      DO KK = 1, 5
        IF(CROP_PER(LCOUNT,KK) .LE. 11.0) X(KK) = 0
      ENDDO

      CALL OP2OPF(FUN, X, XL, XU, NX, NY, LOOP, 900, 0, IERR)
*     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      LOOP = 1
      WT_FACT = 100.0

      DO KK = 1, 5
        IF(CROP_PER(LCOUNT,KK) .LE. 11.0) X(KK) = 0
      ENDDO

      CALL OP2OPF(FUN, X, XL, XU, NX, NY, LOOP, 900, 0, IERR)
*     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      LOOP = 1
      WT_FACT = 10.0

      DO KK = 1, 5
        IF(CROP_PER(LCOUNT,KK) .LE. 11.0) X(KK) = 0
      ENDDO

      CALL OP2OPF(FUN, X, XL, XU, NX, NY, LOOP, 900, 0, IERR)
*     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      LOOP = 1
      WT_FACT = 1.0

      DO KK = 1, 5
        IF(CROP_PER(LCOUNT,KK) .LE. 11.0) X(KK) = 0
      ENDDO

      CALL OP2OPF(FUN, X, XL, XU, NX, NY, LOOP, 900, 0, IERR)
*     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      DO KK = 1, 5
        IF(CROP_PER(LCOUNT,KK) .LE. 0.95) X(KK) = 0
      ENDDO

1001  CONTINUE

      DO KK = 1, 5
        IX(KK) = NINT(X(KK))
         IF (IX(KK) .GT. IMAX(KK)) THEN
          IMAX(KK) = IX(KK)
       ENDIF
      ENDDO
*
*  WE NOW HAVE THE TOTAL NUMBER OF FIELDS FOR CROP "i" STORED IN THE ARRAY
*  IX(I), I = 1 FOR TV, 2 FOR FC, 3 FOR NC, 4 FOR SB, AND 5 FOR PP.  NOTE: WE
*  HAVE ALL RELEVANT, STOCHASTIC INFORMATION IN ARRAYS FOR FIELD AREA, APPLICATION
*  RATE, DATE, AND SO FORTH.  IF FOUR ENTRIES ARE REQUIRED FOR TV FIELDS, THEN,
```

```
*  THE FIRST FOUR ENTRIES FROM THE ARRAYS ARE USED.  FOR THE NEXT TOWNSHIP, IF 6
*  TV ENTRIES ARE REQUIRED, THEN THE NEXT 6 ENTRIES IN THE TV ARRAYS ARE USED.
*  IN THIS WAY, WE MOVE DOWN THE ARRAYS (CONTAINING RANDOM, STOCHASTIC VALUES VIA
*  CRYSTAL BALL SAMPLING OF THE USER DEFINED PDF'S) SUCH THAT DIFFERENT RANDOM
*  SELECTIONS ARE ALWAYS USED.  LOCATIONS WITHIN THESE DATA ARRAYS ARE GIVEN BY
*  IX_OLD AND IX_NEW(I).  ALSO, REMEMBER THAT X(I) IS REAL AND THUS WE MUST
*  CONVERT TO AN INTEGER FOR SUMMATION.


      DO I = 1, 5
         IF( NINT(X(I)) .EQ. 0) THEN
           IUPPER = IX_OLD(I)
         ELSE
           IUPPER = IX_OLD(I) + IX(I) - 1
         ENDIF

*        PRINT *, IX_OLD(I),IUPPER, IX(I)
      ENDDO


*     ===========
      DO I = 1, 5
*     ===========

         IF( NINT(X(I)) .EQ. 0) THEN
           IUPPER = IX_OLD(I)-1
         ELSE
           IUPPER = IX_OLD(I) + IX(I) - 1
         ENDIF

         DO L = IX_OLD(I), IUPPER
            KTOT = KTOT + 1
            X_AREA(KTOT) = AREA(I,L)
            X_RATE(KTOT) = RATE(I,L)
            X_AF(KTOT) = CMASS(I,L)/AMASS(I,L)
            X_DEPTH(KTOT) = DEPTH(I,L)
            JULDATE(KTOT) = IDATE(I,L)
            DS(KTOT) = D_or_S(I,L)
            TARP(KTOT) = ITARP(I,L)
            SF_INCRP(KTOT) = SINCORP(I,L)
            SF_YR(KTOT) = SYR(I,L)
         ENDDO

        IX_OLD(I) = IUPPER+1

*     ===========
      ENDDO
*     ===========
*
      ITOT_FLDS(LCOUNT) = KTOT

      LOOP = 1

      IF(LOOP .EQ. 1) THEN
       EPSP=1.0E-8
        EPSF=1.0E-6
         STEP=1.0
         ALPHA=1.0
         BETA=0.5
       GAMMA=2.0
      ENDIF

      NX1 = KTOT
      NY1 = 0

* DEFINE LOWER AND UPPER BOUNDS FOR FIELD AREA.  CURRENLTY, ONLY ALLOW FIELD
* SHRINKING/STRETCHING TO BE ~ 10% OF ORIGINAL VALUE VIA CRYSTAL BALL SAMPLING.

      DO K = 1, KTOT
        XL_AREA(K) = X_AREA(K)*.8
        XU_AREA(K) = X_AREA(K)*1.2
```

```
        X_AREA_START(K) = X_AREA(K)
       ENDDO

       WT_FACT = 0.001
       CALL OPTIMIZ(FUN, X_AREA, XL_AREA, XU_AREA, NX1, NY1,
      1             LOOP, 900, 0, IERR)

       LOOP = 1
       WT_FACT = 0.01
       CALL OPTIMIZ(FUN, X_AREA, XL_AREA, XU_AREA, NX1, NY1,
      1             LOOP, 900, 0, IERR)

* Add any new fields with repeat fields to obtain totals.  Note: field
* parameters are in single dimension arrays from 1 to KTOT.  Here, KTOT
* equals total number of fields being written to the optimization output
* file.  Ktot = sum IX(I), I=1,5.  Thus, we know enough information to
* add in any "repeat" field information to the appropriate arrays before
* final printing to the output file "field_opt.dat".  It is this output
* file that is used in the Excel workbook modeling for
       IF (ITERATION .GT. 1) THEN
          DO I = 1, 5
            KX(I) = IX(I) + JX(LCOUNT,I)
          ENDDO
       ELSE
          DO I = 1, 5
            KX(I) = IX(I)
          ENDDO
       ENDIF

       WRITE(*,111)LCOUNT,KX(1),KX(2),KX(3),KX(4),KX(5),RESID3,RESID4
       WRITE(2,112)LCOUNT,KX(1),KX(2),KX(3),KX(4),KX(5)

111    FORMAT(' TOWNSHIP ID#',I1,5i4,2X,F10.2,E11.3)
112    FORMAT(I3,5I5)


       DO I = 1, 5
         IX_OLD(I) = 1
       ENDDO

       X0 = 0
       Y0 = 0
       Z0 = 0
       L = 0
*++++++++++++++++++
       DO I = 1, 5
*++++++++++++++++++
*      FIRST, WRITE OUT THE REPEAT FIELD INFORMATION IF THE YEAR OF SIMULATION IS
*      > 1 (i.e., for the first year of simulation, there are no "repeat" fields.
*      ^^^^^^^^^^^^^^^^^^^^^^^^^^
       IF (ITERATION .GT. 1) THEN
*      ^^^^^^^^^^^^^^^^^^^^^^^^^^
*
         FR = 1
*        ''''''''''''''''''''''''
         DO KK = 1, JX(LCOUNT,I)
*        ''''''''''''''''''''''''
          WRITE(2,*)LCOUNT, AREA_R(LCOUNT,I,KK),
     #    AREA_R(LCOUNT,I,KK),RATE_R(LCOUNT,I,KK),
     #    AF_R(LCOUNT,I,KK), DEPTH_R(LCOUNT,I,KK),
     #    IDATE_R(LCOUNT,I,KK), D_or_S_R(LCOUNT,I,KK),
     #    ITARP_R(LCOUNT,I,KK), SINCORP_R(LCOUNT,I,KK),
     #    SYR_R(LCOUNT,I,KK),FR,X_R(LCOUNT,I,KK),
     #    Y_R(LCOUNT,I,KK),Z_R(LCOUNT,I,KK)
*        ''''''''''''''''''''''''
         ENDDO
*        ''''''''''''''''''''''''
*
*      ^^^^^^^^^^^^^^^^^^^^^^^^^^
       ENDIF
*      ^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
*  UPDATE WHERE IN THE ARRAYS FROM "FIELD.DAT" TO GET FIELD INFORMATION FOR
*  THE NEXT TOWNSHIP.


*      NOW, WRITE OUT THE NEW FIELD INFORMATION

      FR = 0
*     ......................
       DO JP = 1, IX(I)
*     ......................
       L = L + 1
*
        WRITE(2,*)LCOUNT, X_AREA_START(L), X_AREA(L),
     #  X_RATE(L), X_AF(L), X_DEPTH(L), JULDATE(L),
     #  DS(L),TARP(L),SF_INCRP(L), SF_YR(L), FR,X0,Y0,Z0
*     ......................
      ENDDO
*     ......................

      IX_OLD(I) = IUPPER + 1

      IF( IX(I) .EQ. 0) THEN
        IUPPER = IX_OLD(I)-1
      ELSE
        IUPPER = IX_OLD(I) + IX(I) - 1
      ENDIF
*++++++++++++++++++
      ENDDO                ! This is the return for CROP TYPE counter
*++++++++++++++++++
*
*===================#########
      ENDDO                ! This is the return for Township ID counter
*===================#########

*      PAUSE
      CLOSE(2)
121   FORMAT(1X, 4I10)
      END

*======================================================================

      SUBROUTINE DWOBJF(FUN, X, NY, NX)
      IMPLICIT REAL*8(A-H, O-Z)
      PARAMETER(MAX=800, MAX1 = 80)

      COMMON /MISC/ CROP_PER(9,5),AREA(5,MAX),RATE(5,MAX),
     #       DEPTH(5,MAX),IDATE(5,MAX),AMASS(5,MAX),CMASS(5,MAX),
     #       LCOUNT,TWN_CAP, FRAC_CAP(9),WT_FACT,IX_OLD(20),
     #       RESID1,RESID2,IX_NEW(20),ITARP(5,MAX),ITERATION

      COMMON /MISC2/ AREA_R(9,5,MAX1), RATE_R(9,5,MAX1),
     1 DEPTH_R(9,5,MAX1), IDATE_R(9,5,MAX1), AMASS_R(9,5,MAX1),
     2 CMASS_R(9,5,MAX1), ITARP_R(9,5,MAX1), X_R(9,5,MAX1),
     3 Y_R(9,5,MAX1), Z_R(9,5,MAX1), JX, AF_R(9,5,MAX1)

      REAL*8 FUN(NX), X(NX),C_FRAC(5)
      INTEGER JX(9,5)

      SUM1 = 0.0
      SUM2 = 0.0

      DO I = 1, 5
        C_FRAC(I)=0
      ENDDO


*     ===========
      DO I = 1, 5
*     ===========
         IF( NINT(X(I)) .EQ. 0) THEN
           IUPPER = 0
```

```
         ELSE
           IUPPER = IX_OLD(I) + NINT(X(I))  - 1
         ENDIF

         DO K = IX_OLD(I), IUPPER
           SUM1 = SUM1 + CMASS(I,K)       !SUMMATION FOR TOTAL KG 1,3-D (ADJUSTED)
           SUM2 = SUM2 + AREA(I,K)        !SUMMATION FOR TOTAL AREA FOR ALL CROP TYPES
           C_FRAC(I) = C_FRAC(I) + AREA(I,K)
         ENDDO
*---------------------------------
* ADD THE RETREATED FIELD AREA
*  AND MASS(ONLY IF ITERATION > 1)

        IF(ITERATION .GT. 1) THEN
          DO KK = 1, JX(LCOUNT,I)
            SUM1 = SUM1 + CMASS_R(LCOUNT,I,KK)
              SUM2 = SUM2 + AREA_R(LCOUNT, I, KK)
            C_FRAC(I) = C_FRAC(I) + AREA_R(LCOUNT, I, KK)
          ENDDO

        ENDIF
*---------------------------------

*     ===========
      ENDDO
*     ===========



      DO I = 1, 5
        C_FRAC(I) = C_FRAC(I)/SUM2*100.
      ENDDO

      RESID1 = 0.0
      RESID2 = 0.0

      DO I = 1, 5
        ENTRY = ABS(C_FRAC(I) - CROP_PER(LCOUNT,I))
        RESID1 = RESID1 + ENTRY
      ENDDO

      RESID2 = abs(TWN_CAP*FRAC_CAP(LCOUNT)-SUM1)

*
       FUN(1) = WT_FACT*RESID1 + RESID2
*
      RETURN
      END
*

*================================================================
      SUBROUTINE OBJFUNC2(FUN, X, NY, NX)
      IMPLICIT REAL*8(A-H, O-Z)
      PARAMETER(MAX=800, MAX1=80)

      COMMON /MISC/ CROP_PER(9,5),AREA(5,MAX),RATE(5,MAX),
     #       DEPTH(5,MAX),IDATE(5,MAX),AMASS(5,MAX),CMASS(5,MAX),
     #       LCOUNT,TWN_CAP, FRAC_CAP(9),WT_FACT,IX_OLD(20),
     #       RESID1,RESID2,IX_NEW(20),ITARP(5,MAX),ITERATION

      COMMON /MISC1/ X_RATE(MAX), X_AF(MAX), IX(20), RESID3, RESID4

      COMMON /MISC2/ AREA_R(9,5,MAX1), RATE_R(9,5,MAX1),
     1 DEPTH_R(9,5,MAX1), IDATE_R(9,5,MAX1), AMASS_R(9,5,MAX1),
     2 CMASS_R(9,5,MAX1), ITARP_R(9,5,MAX1), X_R(9,5,MAX1),
     3 Y_R(9,5,MAX1), Z_R(9,5,MAX1), JX, AF_R(9,5,MAX1)

      REAL*8 FUN(NX), X(NX), C_FRAC(MAX)
      INTEGER JX(9,5)

      SUM1 = 0.0
```

```
      SUM2 = 0.0
      RESID3 = 0.0
      RESID4 = 0.0

* Total current "corrected" mass
      DO K = 1, NX
        SUM1 = SUM1 +  X(K)*X_RATE(K)*X_AF(K)
      ENDDO

*      PRINT *, 'NX AND SUM1 =  ', NX, SUM1, TWN_CAP*FRAC_CAP(LCOUNT)

      DO I = 1, 5
        C_FRAC(I)=0
      ENDDO

      JOLD = 1


      KTOT = 0
*     ============
      DO I = 1, 5
*     ============
         IF( IX(I) .EQ. 0) THEN
           IUPPER = 0
         ELSE
           IUPPER = JOLD + IX(I)  - 1
         ENDIF

         DO K = JOLD, IUPPER
         KTOT = KTOT + 1
           SUM2 = SUM2 + X(KTOT)            !SUMMATION FOR TOTAL AREA FOR ALL CROP TYPES
           C_FRAC(I) = C_FRAC(I) + X(KTOT)
         ENDDO
      JOLD = IX(I) + 1


*---------------------------------
* ADD THE RETREATED FIELD AREA
*  AND MASS(ONLY IF ITERATION > 1)

        IF(ITERATION .GT. 1) THEN
          DO KK = 1, JX(LCOUNT,I)
              SUM2 = SUM2 + AREA_R(LCOUNT, I, KK)
              SUM1 = SUM1 + CMASS_R(LCOUNT, I, KK)
            C_FRAC(I) = C_FRAC(I) + AREA_R(LCOUNT, I, KK)
           ENDDO

        ENDIF
*---------------------------------

*     ============
      ENDDO
*     ============

      DO I = 1, 5
        C_FRAC(I) = C_FRAC(I)/SUM2*100.
      ENDDO


      DO I = 1, 5
        ENTRY = ABS(C_FRAC(I) - CROP_PER(LCOUNT,I))
        RESID3 = RESID3 + ENTRY
      ENDDO

      RESID4 = ABS(TWN_CAP*FRAC_CAP(LCOUNT) - SUM1)

      FUN(1) = RESID3 + WT_FACT*RESID4

*
      RETURN
      END
```

```
*==========================================================================
      SUBROUTINE OP2OPF(FUNCT, X, XL, XU, NX, NY, LOOP, MAXIT, IPRINT,
     1 IERR)
C
C ***********************************************************************
C     COPYRIGHT (C) 1992 BY THE DOW CHEMICAL COMPANY, MIDLAND, MICHIGAN
C ***********************************************************************
C
C     MODIFIED FLEXIBLE POLYGON SEARCH ROUTINE TO FIND THE MINIMUM OF A
C     FUNCTION WITH PARAMETER AND OTHER VARIABLES CONSTRAINED
C
C     DWOBJF = EXTERNAL SUBROUTINE THAT CALCULATES THE OBJECTIVE
C              FUNCTION VALUE (FUNCT(1)) AND CONSTRAINTED FUNCTIONS
C              (FUNCT(2 TO NY+1)) FROM CURRENT PARAMETER
C              VALUES (X(1-NX)).
C              SUBROUTINE CALLING SEQUENCE SHOULD BE:
C                  CALL DWOBJF(FUNCT, X, NX+NY, NX)
C              FUNCT AND X ARE DOUBLE PRECISION VECTORS AND NX AND
C              NY ARE ONE-WORD INTEGERS.
C     FUNCT  = VECTOR CONTAINING FUNCTION VALUE (FIRST LOCATION) AND
C              CONSTRAINED VARIALBES (2 TO NY + 1) (OUTPUT)
C     X      = VECTOR OF 'NX' PARAMETER VALUES (UPDATE)
C     XL     = VECTOR OF 'NX' LOWER PARAMETER CONSTRAINT VALUES AND
C              'NY' LOWER CONSTRAINT FUNCTION BOUNDS (INPUT).
C     XU     = VECTOR OF 'NX' UPPER PARAMETER CONSTRAINT VALUES AND
C              'NY' UPPER CONSTRAINT FUNCTION BOUNDS (INPUT).
C     NX     = NUMBER OF PARAMETERS IN X VECTOR (INPUT).
C              A NEGATIVE VALUES INDICATES UNCONSTRAINED OPTIMIZATION
C              WITH XU AND XL VECTORS NOT TO BE USED.
C     NY     = NUMBER OF CONSTRAINT FUNCTIONS (INPUT).
C     LOOP   = NUMBER OF TRIALS (UPDATE).
C            = 0 FOR INITIAL ENTRY WHEN DEFAULT COEFFICIENTS VALUES
C                ARE TO BE USED.
C            = 1 FOR INITIAL ENTRY WHEN COEFFICIENT VALUES SET BY THE
C                USER IN LABELLED COMMON ARE TO BE USED:
C                COMMON /FITCOM/ EPSP, EPSF, STEP, ALPHA, BETA, GAMMA
C                THE VALUES THAT MUST BE SET ARE:
C                EPSP  = CONVERGENCE IN PARAMETER VALUES(1.E-8 ASSUMED)
C                EPSF  = CONVERGENCE IN FUNCTION VALUES (1.E-8 ASSUMED)
C                STEP  = INITIAL FRACTIONAL STEP SIZE     (.20 ASSUMED)
C                ALPHA = REFLECTION COEFFICIENT           (1.0 ASSUMED)
C                BETA  = CONTRACTION COEFFICIENT          (.5 ASSUMED)
C                GAMMA = EXPANSION COEFFICIENT            (2.0 ASSUMED)
C            NOTE: ALL THESE VALUES ARE DOUBLE PRECISION.
C
C     MAXIT  = MAXIMUM NUMBER OF ITERATIONS (INPUT).
C     IPRINT = 0 FOR NO PRINTING (INPUT).
C            = WRITE UNIT NUMBER FOR PRINTING OF ALL TRIALS
C            = - WRITE UNIT NUMBER FOR PRINTING OF FUNCTION IMPROVEMENTS
C     IERR   = RETURN INDICATOR (OUTPUT).
C            =  0 FOR CONVERGED WITH NO ERRORS
C            =  1 FOR FAILED TO CONVERGE IN MAXIT TRIALS
C            =  2 FOR MORE THAN 20 PARAMETERS PLUS OTHER VARIABLES
C                SPECIFIED
C            =  J FOR PARAMETER X(J-2) IS OUT OF BOUNDS
C
C   EXAMPLE:
C     REAL*8 X( ), XL( ), XU( )
C     SET NX, XL, XU VALUES AND STARTING ESTIMATES FOR X
C     LOOP    = 0
C     CALL OP2CPF( DWOBJF, FUN, X, XL, XU, NX, 2, LOOP, 300, 0, IERR )
C     IF ( IERR ) 10, 20, 10
C  10 PROCESS ERRORS
C  20 CONTINUE IN PROGRAM
C     ...
C    END
C    SUBROUTINE DWOBJF ( FUN, X, NY, NX )
C    REAL*8 FUN(1), X(1)
C    FUN(1) = FUNCTION OF X VECTOR FOR OBJECTIVE FUNCTION
C    FUN(2) = FUNCTION OF X VECTOR FOR CONSTRAINTS
```

```
C     ...
C    RETURN
C    END
C
C
C   REFERENCE:
C    HIMMELBLAU, D.M., 'APPLIED NONLINEAR PROGRAMMING,'
C    MCGRAW-HILL, NEW YORK (1972).
C
C *********************************************************************
C
      IMPLICIT REAL*8(A-H, O-Z)
      REAL*8 FUNCT(1), X(1), XL(1), XU(1)
      COMMON /FITCOM/ EPSP, EPSF, STEP, ALPHA, BETA, GAMMA

*      REAL*8 FUNCT(1)
*      INTEGER X(1), XL(1), XU(1)
C
C
*      REAL*8 D(21, 26), FX(26), SCALE(20)
      REAL*8 D(51, 56), FX(56), SCALE(50)
      INTEGER FIND, PATH
C
      CHARACTER*8 PHASE(8)

      DATA PHASE/'Initial ', 'Vertex  ', 'Vertex  ', 'Reflect ',
     1 'Expand  ', 'Contract', 'Reduce  ', 'Solution'/

C
C     SET INITIAL VALUES OF ROUTINE CONSTANTS
C
      IF (LOOP .NE. 1) THEN
         IF (LOOP .GT. 1) GOTO 90
         ALPHA = 1.
         BETA = .5
         GAMMA = 2.0
         EPSP = 1.D-6
         EPSF = 1.D-6
         STEP = 0.01
         LOOP = 1
      ENDIF


      PATH = 1
      FIND = 1
      N = IABS(NX)
      NX1 = N + 1
      NPTS = N + 1
      IERR = 2
      IF (NY .GT. 19) GOTO 581
      IF (NX .GT. 50) GOTO 581
      AN = N
      LAST = N + 2
      ALAST = LAST
      ALSTM1 = LAST - 1
      MEAN = LAST + 1
      NR = MEAN + 1
      NE = NR + 1
      NC = NE + 1
      NCDONE = 0
      MOVES = 0
      IF (NX .LT. 0) GOTO 90
      IF (NX .NE. 0) THEN
         DO I = 1, N
            IF (X(I) .LT. XL(I)) GOTO 80
            IF (X(I) .GT. XU(I)) GOTO 80
         ENDDO
      ENDIF
      IF (NY .LE. 0) GOTO 90
      CALL DWOBJF(FUNCT, X, 1 + NY, NX)
      DO I = 1, NY
```

```
         IF (FUNCT(I + 1) .LT. XL(I + N)) GOTO 80
         IF (FUNCT(I + 1) .GT. XU(I + N)) GOTO 80
      ENDDO
   92 IF (IPRINT .NE. 0) THEN
         IF (IPRINT .LE. 0) THEN
            IF (LOOP .NE. 1) THEN
               IF (FUNCT(1) .GT. FX(MIN)) GOTO 110
            ENDIF
         ENDIF
         IPRNT = IABS(IPRINT)
         WRITE (IPRNT, 580) FUNCT(1), PHASE(PATH), LOOP, (X(I), I = 1,
     1    N)
  580    FORMAT (/, ' Function =', 1P, G12.5, ' - ', A8, ' after', I4,
     1    ' Trials at:', (/, 1P, 6G12.5))
      ENDIF
C
C    RETURN IF SOLUTION FUNCTION VALUES WERE JUST CALCULATED
C
  110 IERR = 0
      IF (MAXIT .EQ. 1) GOTO 581
      IF (LOOP .GE. MAXIT) ITERM = 3
      IF (PATH .EQ. 8) GOTO 581
C
C    CHECK IF OTHER VARIABLES RANGES HAVE BEEN EXCEEDED
C
      IF (LOOP .NE. 1) THEN
         IF (NY .GT. 0) THEN
            IF (MOVES .LT. 10) THEN
               DO I = 1, NY
                  VALUE = FUNCT(I + 1)
                  IF (VALUE .LT. XL(I + N)) GOTO 150
                  IF (XU(I + N) .LT. VALUE) GOTO 150
               ENDDO
               GOTO 170
  150          CONTINUE
               DO I = 1, N
                  D(I, FIND) = D(I, MEAN) + .5*(D(I, FIND) - D(I, MEAN)
     1             )
               ENDDO
               MOVES = MOVES + 1
               GOTO 520
            ELSEIF (MOVES .LE. 10) THEN
               DO I = 1, N
                  D(I, FIND) = D(I, MEAN)
               ENDDO
               MOVES = MOVES + 1
               GOTO 520
            ENDIF
         ENDIF
      ENDIF
C
C    CHECK IF NUMBER OF TRIALS MAXIMUM HAS BEEN EXCEEDED
C
  170 MOVE = MOVES
      MOVES = 0
      LOOP = LOOP + 1
      FX(FIND) = FUNCT(1)
      IF (LOOP .GE. MAXIT) THEN
         IF (LOOP - N .GE. 2) THEN
            FIND = MIN
            PATH = 8
            ITERM = 3
            GOTO 520
         ENDIF
      ENDIF
C
      IF (PATH .NE. 2) THEN
         IF (PATH .NE. 3) THEN
            IF (PATH .EQ. 4) THEN
C
C    PATH 4  -  TEST FX(NR) VALUE AND EXPAND OR CONTRACT
```

```
C
               IF (FX(NR) .GT. FMIN) THEN
C
                  DO I = 1, LAST
                     IF (I .NE. MAX) THEN
                        IF (FX(NR) .LT. FX(I)) GOTO 400
                     ENDIF
                  ENDDO
                  IF (FX(NR) .LE. FMAX) THEN
                     FX(MAX) = FX(NR)
                     DO I = 1, N
                        D(I, MAX) = D(I, NR)
                     ENDDO
                  ENDIF
C
C     CONTRACTION CALCULATIONS
C
                  DO I = 1, N
                     D(I, NC) = D(I, MEAN) + BETA*(D(I, MAX) - D(I,
     1                MEAN))
                  ENDDO
                  PATH = 6
                  FIND = NC
                  NCDONE = 1
                  GOTO 520
C
C     EXPANSION CALCULATIONS
C
               ELSEIF (MOVE .LE. 0) THEN
                  DO I = 1, N
                     D(I, NE) = D(I, MEAN) + GAMMA*(D(I, NR) - D(I,
     1                MEAN))
                  ENDDO
                  FIND = NE
                  PATH = 5
                  GOTO 520
               ENDIF
            ELSEIF (PATH .EQ. 5) THEN
C
C     PATH 5  -  TEST RESULTS OF EXPANSION CALCULATION
C
               IF (FX(NE) .LT. FMIN) THEN
                  FIND = 0
                  ISAVE = NE
                  IF (FX(NE) .GT. FX(NR)) ISAVE = NR
                  FX(MAX) = FX(ISAVE)
                  DO I = 1, N
                     D(I, MAX) = D(I, ISAVE)
                  ENDDO
                  ISAVE = NE
                  IF (FX(NE) .LE. FX(NR)) ISAVE = NR
                  GOTO 490
               ENDIF
            ELSEIF (PATH .EQ. 6) THEN
C
C     PATH 6  -  TEST RESULTS OF CONTRACT CALCULATION AND REDUCE
C
               IF (FX(NC) .LT. FMAX) THEN
C
                  FX(MAX) = FX(NC)
                  DO I = 1, N
                     D(I, MAX) = D(I, NC)
                  ENDDO
                  FIND = 0
                  GOTO 490
               ELSE
C
C     REDUCTION CALCULATIONS
C
                  DO I = 1, N
                     TEMP = D(I, 1)
```

```
                     D(I, 1) = D(I, MIN)
                     D(I, MIN) = TEMP
                  ENDDO
                  TEMP = FX(1)
                  FX(1) = FX(MIN)
                  FX(MIN) = TEMP
                  DO I = 2, LAST
                     DO J = 1, N
                        D(J, I) = 0.5*(D(J, 1) + D(J, I))
                     ENDDO
                  ENDDO
C
C    RESCALE PROBLEM AS PART OF REDUCTION
C
                  DO I = 2, LAST
                     DO J = 1, N
                        D(J, I) = D(J, I)*SCALE(J)
                     ENDDO
                  ENDDO
C
                  DO I = 1, N
                     SCALE(I) = D(I, 1)*SCALE(I)
                     D(I, 1) = 1.
                     IF (SCALE(I) .EQ. 0.) THEN
                        D(I, 1) = 0.
                        SCALE(I) = 1.
                     ENDIF
                  ENDDO
C
                  DO I = 2, LAST
                     DO J = 1, N
                        D(J, I) = D(J, I)/SCALE(J)
                     ENDDO
                  ENDDO
C
                  PATH = 7
                  FIND = 1
                  GOTO 520
               ENDIF
            ELSEIF (PATH .EQ. 7) THEN
               GOTO 250
            ELSE
C
C    PATH 1  -  SET SEARCH CONSTANTS AND SCALE PROBLEM
C
               FIND = 1
               PATH = 2
C
               DO I = 1, N
                  SCALE(I) = X(I)
                  D(I, 1) = 1.
                  D(I, MEAN) = 1.
                  IF (SCALE(I) .EQ. 0.) THEN
                     SCALE(I) = 1.
                     D(I, 1) = 0.
                     D(I, MEAN) = 0.
                  ENDIF
               ENDDO
C
               D1 = .7071*STEP*(DSQRT(AN + 1.) + AN - 1.)/AN
               D2 = .7071*STEP*(DSQRT(AN + 1.) - 1.)/AN
               GOTO 210
            ENDIF
C
  400       FX(MAX) = FX(NR)
            DO I = 1, N
               D(I, MAX) = D(I, NR)
            ENDDO
            FIND = 0
C
C    TEST FOR CONVERGENCE
```

```
C
C    STANDARD DEVIATION OF FUNCTION VALUES
  490       XMEAN = 0.
            DO J = 1, LAST
               XMEAN = XMEAN + FX(J)
            ENDDO
            XMEAN = XMEAN/ALAST
            XVAR = 0.
            DO J = 1, LAST
               XVAR = XVAR + (FX(J) - XMEAN)**2
            ENDDO
            XVAR = XVAR/ALAST
            XSD = DSQRT(XVAR)/DABS(XMEAN)
            IF (XSD .LE. EPSF) ITERM = 1
            IF (XSD .GT. EPSF) THEN
C
C     STANDARD DEVIATION OF SIMPLEX VERTICE
               DO I = 1, N
                  DMEAN = 0.
                  DO J = 1, LAST
                     DMEAN = DMEAN + D(I, J)
                  ENDDO
                  DMEAN = DMEAN/ALAST
                  DVAR = 0.
                  DO J = 1, LAST
                     DVAR = DVAR + (D(I, J) - DMEAN)**2
                  ENDDO
                  DVAR = DVAR/ALAST
                  DSD = DSQRT(DVAR)/DABS(DMEAN)
                  IF (DSD .GT. EPSP) GOTO 510
                  ITERM = 2
               ENDDO
               GOTO 505
  510          IF (FIND .EQ. 0) GOTO 260
               GOTO 520
            ENDIF
C
  505       FIND = MIN
            IF ((NCDONE .EQ. 1) .AND. (FX(NC) .LT. FX(MIN))) FIND = NC
            PATH = 8
            GOTO 520
         ENDIF
C
C    PATH 3  -  FIND MINIMUM AND MAXIMUM VALUES AND FIND CENTROID
C
  260    FMAX = FX(1)
         MAX = 1
         FMIN = FX(1)
         MIN = 1
         DO I = 2, LAST
            IF (FX(I) .GT. FMAX) THEN
               FMAX = FX(I)
               MAX = I
            ELSEIF (FX(I) .LT. FMIN) THEN
               FMIN = FX(I)
               MIN = I
            ENDIF
         ENDDO
C
C    CENTROID CALCULATION
C
         DO I = 1, N
            SUM = 0
            DO J = 1, LAST
               SUM = SUM + D(I, J)
            ENDDO
            D(I, MEAN) = (SUM - D(I, MAX))/ALSTM1
         ENDDO
C
C    REFLECT MAXIMUM THRU THE CENTROID
C
```

```
         DO I = 1, N
            D(I, NR) = D(I, MEAN) + ALPHA*(D(I, MEAN) - D(I, MAX))
         ENDDO
         PATH = 4
         FIND = NR
         GOTO 520
      ENDIF
C
C    PATH 2  -  SETUP VERTEX VALUES AROUND INITIAL GUESS
C
  210 IF (FIND .GT. N + 1) THEN
         J1 = FIND - N + 1
         DO I = 1, N
            D(I, FIND + 1) = 2.*D(I, 1) - D(I, J1)
         ENDDO
      ELSE
         DO I = 1, N
            D(I, FIND + 1) = D(I, 1) + D2
         ENDDO
         D(FIND, FIND + 1) = D(FIND, 1) + D1
      ENDIF
  250 FIND = FIND + 1
      IF (FIND .GE. LAST) PATH = 3
C
C     SETUP PARAMETERS FOR NEXT TRIAL AND CHECK THEIR RANGES
C
  520 CONTINUE
      DO I = 1, N
         X(I) = D(I, FIND)*SCALE(I)
         IF (NX .GT. 0) THEN
            IF (XU(I) .LE. X(I)) THEN
               X(I) = XU(I) - .1*(XU(I) - D(I, MEAN)*SCALE(I))
               D(I, FIND) = X(I)/SCALE(I)
            ENDIF
            IF (X(I) .LE. XL(I)) THEN
               X(I) = XL(I) - .1*(XL(I) - D(I, MEAN)*SCALE(I))
               D(I, FIND) = X(I)/SCALE(I)
            ENDIF
         ENDIF
      ENDDO
C
C    PRINT FUNCTION AND PARAMETERS VALUES IF DESIRED
C
   90 CALL DWOBJF(FUNCT, X, 1 + NY, NX)
      GOTO 92
   80 IERR = I + 2
  581 RETURN
C
      END


*=====================================================================
      SUBROUTINE OPTIMIZ(FUNCT, X_AREA, XL_AREA, XU_AREA, NX1, NY1,
     1                   LOOP, MAXIT, IPRINT, IERR)
C
C  SAME AS SUBROUTINE OP2OPF EXCEPT THIS ROUTINE CALLS THE SUBROUTINE
C  OBJFUNC2 WHICH CONTAINS THE OBJECTIVE FUNCTION FOR STRETCHING/SHRINKING
C  THE PREDETERMINED FIELD SIZES TO MEET CONSTRAINTS.
C
C **********************************************************************
C
      IMPLICIT REAL*8(A-H, O-Z)
      REAL*8 FUNCT(1), X_AREA(1), XL_AREA(1), XU_AREA(1)
      COMMON /FITCOM/ EPSP, EPSF, STEP, ALPHA, BETA, GAMMA

C
*      REAL*8 D(21, 26), FX(26), SCALE(20)
      REAL*8 D(71, 76), FX(76), SCALE(70)

      INTEGER FIND, PATH
C
```

```
      CHARACTER*8 PHASE(8)

      DATA PHASE/'Initial ', 'Vertex  ', 'Vertex  ', 'Reflect ',
     1 'Expand  ', 'Contract', 'Reduce  ', 'Solution'/

C
C     SET INITIAL VALUES OF ROUTINE CONSTANTS
C
      IF (LOOP .NE. 1) THEN
         IF (LOOP .GT. 1) GOTO 90
         ALPHA = 1.
         BETA = .5
         GAMMA = 2.0
         EPSP = 1.D-6
         EPSF = 1.D-6
         STEP = 0.01
         LOOP = 1
      ENDIF

      PATH = 1
      FIND = 1
      N = IABS(NX1)
      NX11 = N + 1
      NPTS = N + 1
      IERR = 2
      IF (NY1 .GT. 19) GOTO 581
      IF (NX1 .GT. 70) GOTO 581
      AN = N
      LAST = N + 2
      ALAST = LAST
      ALSTM1 = LAST - 1
      MEAN = LAST + 1
      NR = MEAN + 1
      NE = NR + 1
      NC = NE + 1
      NCDONE = 0
      MOVES = 0
      IF (NX1 .LT. 0) GOTO 90
      IF (NX1 .NE. 0) THEN
         DO I = 1, N
            IF (X_AREA(I) .LT. XL_AREA(I)) GOTO 80
            IF (X_AREA(I) .GT. XU_AREA(I)) GOTO 80
         ENDDO
      ENDIF
      IF (NY1 .LE. 0) GOTO 90
      CALL OBJFUNC2(FUNCT, X_AREA, 1 + NY1, NX1)
      DO I = 1, NY1
         IF (FUNCT(I + 1) .LT. XL_AREA(I + N)) GOTO 80
         IF (FUNCT(I + 1) .GT. XU_AREA(I + N)) GOTO 80
      ENDDO
   92 IF (IPRINT .NE. 0) THEN
         IF (IPRINT .LE. 0) THEN
            IF (LOOP .NE. 1) THEN
               IF (FUNCT(1) .GT. FX(MIN)) GOTO 110
            ENDIF
         ENDIF
         IPRNT = IABS(IPRINT)
         WRITE (IPRNT, 580) FUNCT(1), PHASE(PATH), LOOP,
     1                  (X_AREA(I), I = 1, N)
  580    FORMAT (/, ' Function =', 1P, G12.5, ' - ', A8, ' after', I4,
     1    ' Trials at:', (/, 1P, 6G12.5))
      ENDIF
C
C    RETURN IF SOLUTION FUNCTION VALUES WERE JUST CALCULATED
C
  110 IERR = 0
      IF (MAXIT .EQ. 1) GOTO 581
      IF (LOOP .GE. MAXIT) ITERM = 3
      IF (PATH .EQ. 8) GOTO 581
C
C    CHECK IF OTHER VARIABLES RANGES HAVE BEEN EXCEEDED
```

```
C
      IF (LOOP .NE. 1) THEN
         IF (NY1 .GT. 0) THEN
            IF (MOVES .LT. 10) THEN
               DO I = 1, NY1
                  VALUE = FUNCT(I + 1)
                  IF (VALUE .LT. XL_AREA(I + N)) GOTO 150
                  IF (XU_AREA(I + N) .LT. VALUE) GOTO 150
               ENDDO
               GOTO 170
  150          CONTINUE
               DO I = 1, N
                  D(I, FIND) = D(I, MEAN) + .5*(D(I, FIND) - D(I, MEAN)
     1             )
               ENDDO
               MOVES = MOVES + 1
               GOTO 520
            ELSEIF (MOVES .LE. 10) THEN
               DO I = 1, N
                  D(I, FIND) = D(I, MEAN)
               ENDDO
               MOVES = MOVES + 1
               GOTO 520
            ENDIF
         ENDIF
      ENDIF
C
C    CHECK IF NUMBER OF TRIALS MAXIMUM HAS BEEN EXCEEDED
C
  170 MOVE = MOVES
      MOVES = 0
      LOOP = LOOP + 1
      FX(FIND) = FUNCT(1)
      IF (LOOP .GE. MAXIT) THEN
         IF (LOOP - N .GE. 2) THEN
            FIND = MIN
            PATH = 8
            ITERM = 3
            GOTO 520
         ENDIF
      ENDIF
C
      IF (PATH .NE. 2) THEN
         IF (PATH .NE. 3) THEN
            IF (PATH .EQ. 4) THEN
C
C    PATH 4  -  TEST FX(NR) VALUE AND EXPAND OR CONTRACT
C
               IF (FX(NR) .GT. FMIN) THEN
C
                  DO I = 1, LAST
                     IF (I .NE. MAX) THEN
                        IF (FX(NR) .LT. FX(I)) GOTO 400
                     ENDIF
                  ENDDO
                  IF (FX(NR) .LE. FMAX) THEN
                     FX(MAX) = FX(NR)
                     DO I = 1, N
                        D(I, MAX) = D(I, NR)
                     ENDDO
                  ENDIF
C
C    CONTRACTION CALCULATIONS
C
                  DO I = 1, N
                     D(I, NC) = D(I, MEAN) + BETA*(D(I, MAX) - D(I,
     1                MEAN))
                  ENDDO
                  PATH = 6
                  FIND = NC
                  NCDONE = 1
```

```
                  GOTO 520
C
C     EXPANSION CALCULATIONS
C
               ELSEIF (MOVE .LE. 0) THEN
                  DO I = 1, N
                     D(I, NE) = D(I, MEAN) + GAMMA*(D(I, NR) - D(I,
     1                MEAN))
                  ENDDO
                  FIND = NE
                  PATH = 5
                  GOTO 520
               ENDIF
            ELSEIF (PATH .EQ. 5) THEN
C
C     PATH 5  -  TEST RESULTS OF EXPANSION CALCULATION
C
               IF (FX(NE) .LT. FMIN) THEN
                  FIND = 0
                  ISAVE = NE
                  IF (FX(NE) .GT. FX(NR)) ISAVE = NR
                  FX(MAX) = FX(ISAVE)
                  DO I = 1, N
                     D(I, MAX) = D(I, ISAVE)
                  ENDDO
                  ISAVE = NE
                  IF (FX(NE) .LE. FX(NR)) ISAVE = NR
                  GOTO 490
               ENDIF
            ELSEIF (PATH .EQ. 6) THEN
C
C     PATH 6  -  TEST RESULTS OF CONTRACT CALCULATION AND REDUCE
C
               IF (FX(NC) .LT. FMAX) THEN
C
                  FX(MAX) = FX(NC)
                  DO I = 1, N
                     D(I, MAX) = D(I, NC)
                  ENDDO
                  FIND = 0
                  GOTO 490
               ELSE
C
C     REDUCTION CALCULATIONS
C
                  DO I = 1, N
                     TEMP = D(I, 1)
                     D(I, 1) = D(I, MIN)
                     D(I, MIN) = TEMP
                  ENDDO
                  TEMP = FX(1)
                  FX(1) = FX(MIN)
                  FX(MIN) = TEMP
                  DO I = 2, LAST
                     DO J = 1, N
                        D(J, I) = 0.5*(D(J, 1) + D(J, I))
                     ENDDO
                  ENDDO
C
C     RESCALE PROBLEM AS PART OF REDUCTION
C
                  DO I = 2, LAST
                     DO J = 1, N
                        D(J, I) = D(J, I)*SCALE(J)
                     ENDDO
                  ENDDO
C
                  DO I = 1, N
                     SCALE(I) = D(I, 1)*SCALE(I)
                     D(I, 1) = 1.
                     IF (SCALE(I) .EQ. 0.) THEN
```

```
                        D(I, 1) = 0.
                        SCALE(I) = 1.
                     ENDIF
                  ENDDO
C
                  DO I = 2, LAST
                     DO J = 1, N
                        D(J, I) = D(J, I)/SCALE(J)
                     ENDDO
                  ENDDO
C
                  PATH = 7
                  FIND = 1
                  GOTO 520
               ENDIF
            ELSEIF (PATH .EQ. 7) THEN
               GOTO 250
            ELSE
C
C    PATH 1  -  SET SEARCH CONSTANTS AND SCALE PROBLEM
C
               FIND = 1
               PATH = 2
C
               DO I = 1, N
                  SCALE(I) = X_AREA(I)
                  D(I, 1) = 1.
                  D(I, MEAN) = 1.
                  IF (SCALE(I) .EQ. 0.) THEN
                     SCALE(I) = 1.
                     D(I, 1) = 0.
                     D(I, MEAN) = 0.
                  ENDIF
               ENDDO
C
               D1 = .7071*STEP*(DSQRT(AN + 1.) + AN - 1.)/AN
               D2 = .7071*STEP*(DSQRT(AN + 1.) - 1.)/AN
               GOTO 210
            ENDIF
C
  400       FX(MAX) = FX(NR)
            DO I = 1, N
               D(I, MAX) = D(I, NR)
            ENDDO
            FIND = 0
C
C    TEST FOR CONVERGENCE
C
C    STANDARD DEVIATION OF FUNCTION VALUES
  490       XMEAN = 0.
            DO J = 1, LAST
               XMEAN = XMEAN + FX(J)
            ENDDO
            XMEAN = XMEAN/ALAST
            XVAR = 0.
            DO J = 1, LAST
               XVAR = XVAR + (FX(J) - XMEAN)**2
            ENDDO
            XVAR = XVAR/ALAST
            XSD = DSQRT(XVAR)/DABS(XMEAN)
            IF (XSD .LE. EPSF) ITERM = 1
            IF (XSD .GT. EPSF) THEN
C
C     STANDARD DEVIATION OF SIMPLEX VERTICE
               DO I = 1, N
                  DMEAN = 0.
                  DO J = 1, LAST
                     DMEAN = DMEAN + D(I, J)
                  ENDDO
                  DMEAN = DMEAN/ALAST
                  DVAR = 0.
```

```
                  DO J = 1, LAST
                     DVAR = DVAR + (D(I, J) - DMEAN)**2
                  ENDDO
                  DVAR = DVAR/ALAST
                  DSD = DSQRT(DVAR)/DABS(DMEAN)
                  IF (DSD .GT. EPSP) GOTO 510
                  ITERM = 2
               ENDDO
               GOTO 505
  510          IF (FIND .EQ. 0) GOTO 260
               GOTO 520
            ENDIF
C
  505       FIND = MIN
            IF ((NCDONE .EQ. 1) .AND. (FX(NC) .LT. FX(MIN))) FIND = NC
            PATH = 8
            GOTO 520
         ENDIF
C
C    PATH 3  -  FIND MINIMUM AND MAXIMUM VALUES AND FIND CENTROID
C
  260    FMAX = FX(1)
         MAX = 1
         FMIN = FX(1)
         MIN = 1
         DO I = 2, LAST
            IF (FX(I) .GT. FMAX) THEN
               FMAX = FX(I)
               MAX = I
            ELSEIF (FX(I) .LT. FMIN) THEN
               FMIN = FX(I)
               MIN = I
            ENDIF
         ENDDO
C
C    CENTROID CALCULATION
C
         DO I = 1, N
            SUM = 0
            DO J = 1, LAST
               SUM = SUM + D(I, J)
            ENDDO
            D(I, MEAN) = (SUM - D(I, MAX))/ALSTM1
         ENDDO
C
C    REFLECT MAXIMUM THRU THE CENTROID
C
         DO I = 1, N
            D(I, NR) = D(I, MEAN) + ALPHA*(D(I, MEAN) - D(I, MAX))
         ENDDO
         PATH = 4
         FIND = NR
         GOTO 520
      ENDIF
C
C    PATH 2  -  SETUP VERTEX VALUES AROUND INITIAL GUESS
C
  210 IF (FIND .GT. N + 1) THEN
         J1 = FIND - N + 1
         DO I = 1, N
            D(I, FIND + 1) = 2.*D(I, 1) - D(I, J1)
         ENDDO
      ELSE
         DO I = 1, N
            D(I, FIND + 1) = D(I, 1) + D2
         ENDDO
         D(FIND, FIND + 1) = D(FIND, 1) + D1
      ENDIF
  250 FIND = FIND + 1
      IF (FIND .GE. LAST) PATH = 3
C
```

```
C     SETUP PARAMETERS FOR NEXT TRIAL AND CHECK THEIR RANGES
C
  520 CONTINUE
      DO I = 1, N
         X_AREA(I) = D(I, FIND)*SCALE(I)
         IF (NX1 .GT. 0) THEN
            IF (XU_AREA(I) .LE. X_AREA(I)) THEN
               X_AREA(I) = XU_AREA(I) - .1*(XU_AREA(I)
     1                           - D(I, MEAN)*SCALE(I))
               D(I, FIND) = X_AREA(I)/SCALE(I)
            ENDIF
            IF (X_AREA(I) .LE. XL_AREA(I)) THEN
               X_AREA(I) = XL_AREA(I) - .1*(XL_AREA(I)
     1                           - D(I, MEAN)*SCALE(I))
               D(I, FIND) = X_AREA(I)/SCALE(I)
            ENDIF
         ENDIF
      ENDDO
C
C    PRINT FUNCTION AND PARAMETERS VALUES IF DESIRED
C
   90 CALL OBJFUNC2(FUNCT, X_AREA, 1 + NY1, NX1)
      GOTO 92
   80 IERR = I + 2
  581 RETURN
C
      END
```

Appendix B. Subroutine Definitions

Sub Main_Program()
'VBA system for 1,3-D township-cap modeling for the California Department of Pesticide Regulations (CDPR). A simulation region is broken up into a 3x3 township area, with the "Township" of interest centrally located. Five different crop types are assumed which include: 1) Tree and Vine, 2) Field crops, 3) Nursery Crops, 4) Strawberries, and 5) Post-Plant vines. Any of these crop types can be used as a catch all misc. crop through appropriate input file parameter magnitudes found in the worksheet "PDF Parameters". Summary of worksheets contained in workbook "CA_twn_cap2_gut3.xls", which is an abbreviation for California Township allocation modeling.

Sub Auto_Open()
This subroutine is automatically executed upon opening of the workbook "CA_twn_cap2_gut3.xls". The dialog sheet with the name of the modeling system, developer, and version number is displayed until the user closes the window (Window must be closed before execution of the worksheet macros can occur.

Sub GOTO_BEG()
Activates the worksheets "PDF Parameters" (i.e., goes to this worksheet from somewhere else).

Sub Check_for_Sheet1()
Check for any other Excel workbooks that may be open. Often, Crystal Ball doesn't work properly if multiple workbooks are open. Keep only the 1,3-D system workbook called "CA_twn_cap2_gut3.xls".

Sub Run_ISCST3(func_per_yr_cropID, PctDone, N_yrs, nspace, rec_ht, icyr, w_file_name, sur_air_ID, up_air_ID, region, rate_coeff, pst_name, terrain, iteration, num_srcs, num_srcs_all, anemo_ht, chron_conc, crop_hit, crop_hit_id, recep_all, tot_receptors)
This subroutine generates the ISCST3 input files and runs ISCST3 for a given year of simulation. Since we broke the system down into five crop types, we have to run ISCST3 five times (once for each crop source location and flux file. Flux and source files are already written for this year of simulation (i.e., flux_**.dat, source_**.dat, where ** = TV, FC, NC, SB, or PP)

Sub Get_receptor_info(nspace, rec_ht, terrain, elevation, tot_receptors, location, population, xgrid, ygrid, hgrid, R_type, P_type, E_type, recep_all, k_loop)
This subroutine generates the file "sampler.dat" that is called from the ISCST3 input file via the REceptor pathword "RE INCLUDED sampler.dat". In the file sampler.dat, the file has the syntax as x,y, elevation, flagpole height, where x and y (x, y)are the coordinates of the receptor. terrain = FLAT or ELEV. If equal to FLAT, then we can supply the x,y coordinates and ignore both the elevation and the flag pole height, the latter because the flagpole height is defined via the ISCST3 pathword "CO FLAGPOLE 1.5" in the ISCST3 input file recep_all = integer flag to tell model to either place receptors only in the township of interest (0) or the entire 3x3 township domain (1). Note: we only put receptors in the township of interest to reduce

CPU overhead, and given the fact that the center township has the most surrounding townships (and thus source terms) so it will give the highest exposure predictions. In the assumed geometry, the first township (bottom left, or the SW corner township) at the SW edge is where the origin was defined. Thus, the SW corner of the township of interest has its origin at approximately (9656, 9656), and each side is approximately 9656 m long. Thus, we use this information when determining the receptor locations via the user defined grid spacing. Terrain elevation information is stored in "elevation(twn ID, i,j), where i and j = 1, 100 (row and column). The township of interest is a 6 mi x6 mi square section of land. In this program, the user defines the land type for all townships in the simulation domain. Thus, it is possible to define the land type that each receptor in the township if interest resides upon (i.e., urban, ag-land, mountain, or water). Note: There will not be any source terms in water, mountains, or urban areas, but receptors in these regions will experience fumigant drift concentrations.

Sub Post_process(nspace, ileap, iteration, R_type, town_avg, crop_hit, crop_hit_id, recep_all, tot_receptors, num_srcs_all, Fortran_opt, Num_percen_pts, avg_per, avg_per1, period_max, Num_avg_per)
This subroutine reads in the 24-hr ISCST3 output files for each crop type, and summarizes the concentration information into a single, 24-hr concentration array for the township (i.e., by summing the air concentration for each field type at each receptor and at the same time). 24-hr exceedence probability data for the township of interest is written to the worksheet "24hr_Summary". ISCST3 output summary files containing 24-hr concentration data are 24hr_TV.out, 24hr_FC.out, 24hr_NC.out, 24hr_SB.out, 24hr_TV.out. Header information for 24hr ISCST3 file. Note: when VBS encounters a ",", it assumes this is a line break. The header for the Period output file (yearly in our simulation) is given below. Line 6 contains all of the "commas" and thus this single line is read in as 12 lines of information.

Sub QuickSort_INT(List() As Integer)
Sorts an array using Quick Sort algorithm. Adapted from "Visual Basic Developers Guide" By D.F. Scott. For use with integer arrays only

Sub QuickSort_SINGLE(List() As Single)
Sorts an array using Quick Sort algorithm. Adapted from "Visual Basic Developers Guide" By D.F. Scott. For use with non-integer arrays only

Sub print_chronic(kc2, N_yrs, Chronic_all, Num_percen_pts, ichunk, tot_receptors)
Determine the average annual concentration for the N_yrs of simulation. Plot this value in summary notation as one of the last column entries in the worksheet "Chronic"

Sub format_worksheets()
Subroutine adds lines and shadings to Worksheet labels

Sub print_simulation_avg(NYR_loop, town_avg, kc2, N_yrs, Num_avg_per, xgrid, ygrid, hgrid, R_type, P_type, avg_per, ichunk, tot_receptors, num_srcs_all)
This subroutine writes the user supplied running average period concentations (i.e., 1-day, 15-day) for each receptor to the worksheet "run_avg_twn". In addition, the "N_yrs" running average receptor concentration is also determined

Sub SORTY(icount, conc_all)
This Heap Sort method taken from numerical recipies and converted from Fortran to VBA

Sub shade_columns(kc2, ichunk, Num_avg_per, N_yrs, tot_receptors, ilast, jtot, num_entries, kc4)
Highlight (shade) summary result columns in the worksheet ”Run_avg_twn".

Sub Call_Begin()
This subroutine opens up the form containing the Progress Status bar. The VBA code associated with "UserForm2" is what fires off the subroutine "Main_Program"

Sub UpdateProgress(Pct)
Provides geometry and details for progress bar

Sub Odd_or_Even(Int_num, even_flag)
This subroutine takes and integer value (Int_num) and determines if the integer is odd (even_flag = 0) or even (even_flag = 1).

Sub Read_new_inputs(wt_ID, bufzone, time_bufzone, Fortran_opt, per_nxt_yr, k_loop)
This subroutine reads in new input values for a new temporal "loop". Values read in from this subroutine are found in the worksheet "forecasts". Remember, each "loop" (counter for loop is "k_loop") can have different properties such as if section or random weighting is to be used, section probabilities, township allocation, etc.

Sub write_headers(NYR_loop, forecast_ID)
Write the column headers for the worksheets “Chronic” and 24hr_max”

Sub fld_solve1(Max_No_Twnships, k_read, kp1, iteration, per_nxt_yr, fld_nxt_yr, A_End, A_Start, scal_fac, Odepth, Appdate, O_D_or_S, Oitarp, X_rate, SF_incorp, SF_yr, IX, FLD_R, X_REP, Y_REP, Z_REP, k_loop, I_External, I_twn, Ext_Twn_ID, N_ext_twn_Hits, kline)
This subroutine is the latest for finding the number of fields for each crop type such that the user specified "corrected" township allocation and crop percentages are achieved (or at least the residuals are minimized). This subroutine uses CB to sample appropriate parameters 500 times and generate an ASCII file called "fields.dat" that contains information required by an external FORTRAN optimization program called "opt.exe". For this optimization, the fields for various crop types can vary in size (unlike the VBA optimization where field sizes for each crop type are the same for a given year of simulation).

Sub App_fac(ApDate, ApDepth, AF, itarp, ID_drip_or_shank)
Determines application factors for scaling cumulative flux losses based upon application type (drip or shank), if a tarp is present, and the time of year. Application factors = 1.0 for all soil fumigants, except for 1,3-D if used in the state of California only.

Sub App_fac(ApDate, ApDepth, AF, itarp, ID_drip_or_shank)

Sub Randomize_flds(upperbound, lowerbound, fld_storage)
This routine takes the fields numbers from lowerbount to upper bound and randomizes them. The random numbers are then stored in the array fld_storage

Sub crop_percent_external(**TwID, k_loop, cp1, cp2, cp3, cp4, cp5)**
This subroutine reads the crop percentages for a specific township that is outside of the central 3x3 simulation domain. FOr outside townships, we have a 23x23 domain, where the central 3x3 township is at the center of this 23x23 domain.

Sub Read_Opt_Results(j, TwID, kline, iteration, fld_nxt_yr, tot_fields, Twn_ID, CYES, CNO, NCOL, krepeat, A_Start, A_End, X_rate, scal_fac, Odepth, Appdate, O_D_or_S, Oitarp, SF_incorp, SF_yr, FLD_R, X_REP, Y_REP, Z_REP)
Read in information/solution brought back from call to opt2.exe (optimization program)

Sub LandCover_color(location, jocation, LANDflag, k_loop)
This subroutine reads in the user supplied information from the worksheet "LandCover" and stores it in the array "Location(ID,i,j)". Since this array is an argument in the subroutine, one can call this routine to bring in the array "Location(ID,i,j)" into other subroutines. LANDflag, ELEVflag, and POPflag are integer flags denoting if the user has supplied GIS data for Land cover, Elevation and population, respectively (0 = no data, 1 = GIS data supplied in worksheet "GIS_data".

Sub Pop_plot_Update(k, istart, iend, User_Pop, MaxPop, MinPop, Num_incr, dinc)
This routine updates graphics found in sheet "Population" for simple & crude contour plots (no smoothing) for elevation and population information that is contained in numeric for in the worksheet "GIS_DATA".

Sub Ele_plot_Update(k, istart, iend, User_Ele, MaxEle, MinEle, Num_incr, dinc)
This routine updates graphics found in sheet "Population" for simple & crude contour plots (no smoothing) for elevation and population information that is contained in numeric for in the worksheet "GIS_DATA".

Sub LandCover_update()
This routine is assigned to the button in the worksheet "LandCover" since it has no arguments. This routine performs the same tasks as the subroutine "LandCover_color". Call this latter routine if you need to use the information stored in the array "location(ID,i,j)" that contains info for ag/non-ag land.

Sub Population_update()
This routine is assigned to the button in the worksheet "Population" since it has no arguments.

Sub Elevation_update()
This routine is assigned to the button in the worksheet "Elevation" since it has no arguments.

Sub Get_App_Date(App_date, App_type, itot, iteration, k_loop)
This routine reads in the application date for each field type App_date(i_flg, ic, itot), App_type(i_flg, ic, itot)

Sub Write_flux_files(isrc, Ap_Type, F_scle_Type, Date_type, Rate_type, i_shank, i_drip, Flux_Shank, Flux_Drip, iteration, ileap, jyr)
This subroutine writes the hourly emissions file for an ISCST3 simulation. Five emission files are generated (one for each crop type). Crop types are 1=TV, 2=FC, 3=NC, 4= SB, 5=PP. Open up hourly emission file for writing source strength information. These files are recreated for each year of simulation

Sub leap_year(IYRO, ileap, idays_tot, jyr, icyr)
Determines if weather year is a leap year

Sub Juli2mon(Julian, ileap, IMON, IDAY)
This subroutines takes a Julian day (1-366) and converts it to the appropriate month and year. Arguments that are brought into this routine include the Julian day (Julian) and the leap year flag (ileap = 1 for non-leap year, =2 for leap year)

Sub pop_fld_flux(i, ic, idays_tot, isrc, Ap_Type, F_scle_Type, Date_type, Rate_type, i_shank, i_drip, Flux_Shank, Flux_Drip, Flux_All)
This subroutine populates the array that summarizes the hourly emission rates for each field type for all hrs within a single year

Sub Terrain_elev(ELEVflag, elevation)
ELEVflag = integer flag telling program if GIS data is available (1). If available, then the information will be read from worksheet "GIS_data". If no GIS data is available, then ELEVflag = 0 and information will be read from the worksheet "Elevation".

Sub pop_census(POPflag, population, k_loop)
POPflag = integer flag telling program if GIS data is available (1). If available, then the information will be read from worksheet "GIS_data". If no GIS data is available, then POPflag = 0 and it is assumed population for all grids is zero.

Sub Section_prob(section, prob, i_section, irow, jrow, irow_neighbor, jrow_neighbor, cumul, neighbor, cumul_neighbor, neighbor_TV, section_TV, prob_TV, i_section_TV, irow_TV, jrow_TV, irow_neighbor_TV, jrow_neighbor_TV, cumul_TV, cumul_neighbor_TV, k_loop)
Read in section probability data from coarse grid found in worksheet "Section_prob" for annual crops (everything but T&V)

Sub Sec_refine_prob(section, i_section, irow, jrow, prob, prob_neighbor, neighbor, irow_neighbor, jrow_neighbor, cumul, cumul_neighbor)
This subroutine determines the total number of user defined sections within each township having non-zero probabilities for receiving a treated field. In addition, all surrounding sections

for each user defined non-zero section is summarized for possible "spill-over" should a user defined section "fill-up" with fields. We begin with a non-zero probability section. All surrounding sections are stored in the order they are found using an integer indexing up to "neighbor(ic)", where ic is the township ID. Thus, a "neighboring" section maybe counted more than once if it is a neighbor to more than one user specified non-zero probability. In this way, a neighboring section that neighbors more than one user supplied non-zero prob. section will have a higher probability of being selected. This "higher" probability is accounted for via the cumulative distribution definition (for examp. Assume a section neighbors two user specified non-zero sections. This section may be in location 3 and 9 for all of the neighboring sections "k", where k = 1 to neighbor(ic). Thus, this section can be pick multiple times depending what the random number was for comparing to the cumulative curve. This has the net effect of summing the probabilities for this neighboring section).

Sub Sec_refine_prob_TV(section_TV, i_section_TV, irow_TV, jrow_TV, prob_TV, prob_neighbor_TV, neighbor_TV, irow_neighbor_TV, jrow_neighbor_TV, cumul_TV, cumul_neighbor_TV)
This subroutine assigns section probabilities to the "refined" internal grid system for T&V fields and also determines the neighboring sections that border a non-zero probability section defined by the user. i_section(i) = number of sections within township "i" having non-zero probabilities (i.e., fields can be place here.

Sub Read_sec_prob(section, k_loop)
If section weighting has been chosen, then read in the section probabilities from the worksheet "Section_prob"

Sub Read_sec_prob_graphics(Twn, k)
Worksheet that places section weighting input by user found in "Section_prob" for the central 3x3 townships to appropriate locations in the worksheet "Twn_mass_Wt_Ext". In this way, the inputs get transferred accordingly. Section weighting information is stored in the array Twn(i,j), where i = 1-9 for the central 3x3 townships, and j = 1-36 corresponding to the 36 sections per township.

Sub Read_sec_prob_TV(section_TV, k_loop)
This subroutine reads in the user supplied section probabilities for T&V fields (Lower graphic found in worksheet "Section_prob".

Sub convert_xy(i, ic, elevation, rand1, rand2, sw_x, sw_y, sw_z, kp, N_grid)
Subroutine converts grid nomenclature to (x,y) coordinates for SW corner of field in meters for use in ISCST3 modeling. Note that grid nomenclature is in rows and columns. Thus, the x value is for the column location, and the y value is from the row location. Depending upon which township (i.e., loop ic), then appropriate meters are added such that the origin is at the sw corner of Township # 1. Note: we have to subtract off the number of grids to convert from the NW corner to the SW corner which is what ISCST3 needs.

Sub convert_rand(ic, rand1, rand2, sw_x, sw_y)

Subroutine converts sw corner of field (sw_x, sw_y) to appropriate integer number (rand1, rand2), where rand1 and rand2 are integers from 1 to 100 which are used as the arguments for the storage arrays. This routine is called when a field is repeated from the previous year since we know the (x,y,z) location for the field, but we need to convert it to the appropriate storage location (rand1,rand2) in the storage arrays. Note that grid nomenclature is in rows and columns. Thus, the x value is from the column location, and the y value is from the row location. Depending upon which township (i.e., loop ic), then appropriate meters are added such that the origin is at the sw corner of Township # 1.

Sub Get_Date(ijul_date, ibs, lc, i, ApDate, App_date, ic, time_bufzone, ileap, l1, l2, l3, l4, l5, mon_beg, day_beg, mon_end, day_end)
i is the id location for the field type (i.e., i equals 1 for T&V, i equals 2 for FC, etc). Call routine to bring back the beginning month and day for the application and the ending month and day from which to "turn off" the buffer constraint via the nomenclature/modifications in the CDPR ISCST3 modified model. Brings back mon_beg, day_beg, mon_end, day_end

Sub Fld_placement(FLD_R, X_REP, Y_REP, Z_REP, fld_nxt_yr, A_End, SF_incorp, SF_yr, Appdate, X_rate, IX, scal_fac, Odepth, O_D_or_S, Fld_size, num_flds, Num_Grids, location, elevation, jocation, iteration, lc, app_rate, App_date, App_type, no_flds, i_drip, i_shank, Flux_Drip, Flux_Shank, Percent_Drip, Scale_flg, ileap, jyr, bufzone, time_bufzone, IYRO, j_section, prob, i_section, irow, jrow, irow_neighbor, jrow_neighbor, cumul, neighbor, cumul_neighbor, jp_m, Date_one, Rate_one, Fld_one, sw_x_one, sw_y_one, sw_z_one, itot, i_section_TV, cumul_TV, irow_TV, jrow_TV, Fortran_opt)
This routine places the fields within usable regions of the township when constrained by the section weighting probability. A usable region is one where ag land exists, where the section has a probability > 0.0 of having a treated field, and no T&V application has yet to be made. Once a T&V application is made, this T&V land is omitted from further source terms. Programming Logic is based upon section weighting (and the land type via the array "Location(township ID, i, j)", where Location can have the following attributes. In this routine, the source location information is written to an ASCII file for ISCST3 input. If "j_section = 0, then this is the first time we have called this routine. Thus, any field locations (size and application rate) will remain unchanged from this year forward in keeping with suggestions of B. Johnson (CDPR) {valid only for wt_ID = 1].

Sub Update_loc_array_w_repeat_flds(X_REP, Y_REP, Z_REP, num_flds, Num_Grids, A_End, fld_nxt_yr, FLD_R, location, jocation)
This subroutine updates the location array for the "repeat" fields such that any "new" field for this year of simulation won't use this area

Sub Write_source_file(ic, i, bufzone, mon_beg, day_beg, mon_end, day_end, IRYO, iteration, Fld_size, w_area, sw_x, sw_y, sw_z, scle, zero, f_name, isrc, l1, l2, l3, l4, l5, rand1, rand2, N_grid)
This subroutine writes the SOurce descriptor portion of the ISCST3 input file

Sub write_Misc1(i, lc, isrc, iteration, ic, sw_x, sw_y, sw_z, Fld_size, app_rate, App_date, m_actual, ijul_date, Fortran_opt, ApDate, ApRate)

One of two subroutines that writes summary information about the source fields to the worksheet "Misc"

Sub write_Misc2(AF1, ibs, lc, ApDepth, F_scale, ID_drip_or_shank, CYES, CNO, itarp, m_actual, ijul_date, m_act_sum, m_adj_sum, Fortran_opt)
One of two subroutines that writes summary information about the source fields to the worksheet "Misc"

Sub Get_section_Num(Xbound, Ybound, section, ic, prob, i_section, irow, jrow, cumul, rand1, rand2)
Subroutine selects a section within township "ic" based upon the user defined probability weighting on section (see worksheet "Section_prob"). Information on cumulative probability is given in cumul(ic, 1 to i_section(ic)), where ic = township ID (1 to 9), and i (1 to i_section(ic)) are points in the cumulative distribution. There are a total of 36 sections per township.

Sub Get_section_TV(Xbound, Ybound, section_TV, ic, prob_TV, i_section_TV, irow_TV, jrow_TV, cumul_TV, rand1, rand2)
Subroutine selects a section within township "ic" for annual crops based upon the user defined probability weighting on section (see worksheet "Section_prob"). Information on cumulative probability is given in cumul(ic, 1 to i_section(ic)), where ic = township ID (1 to 9), and i (1 to i_section(ic)) are points in the cumulative distribution. There are a total of 36 sections per township.

Sub Get_section_Num_fit(Xbound, Ybound, section, ic, prob_neighbor, neighbor, irow_neighbor, jrow_neighbor, cumul_neighbor, rand1, rand2)
Subroutine selects a section within township "ic" based upon the user defined probability weighting on section (see worksheet "Section_prob"). Information on cumulative probability is given in cumul(ic, 1 to i_section(ic)), where ic = township ID (1 to 9), and i (1 to i_section(ic)) are points in the cumulative distribution. There are a total of 36 sections per township.

Sub check_if_fld_fits(ic, rand1, rand2, N_grid, location, jocation, fld_used, Fortran_opt)
This routine looks to see if a field can "fit" at the given location, where the SW corner (via rand1 and rand2) and the field area (via N_grid) are known. If the flag "fld_used" equals 1, then the field at this location won't fit. If fld_used = 0, then a field can fit.

Sub Store_first_year_src(ic, i, i2, isrc, Ap_Type, F_scale, ApDate, ApRate, Fld_size, w_area, sw_x, sw_y, sw_z, Date_one, Rate_one, Fld_one, sw_x_one, sw_y_one, sw_z_one)
This routine stores information about the source size, location, etc. from the first year of simulation for a "section" weighted simulation.

Sub New_TV_location(ic, iteration, i_section, i_section_TV, sw_x, sw_y, sw_z, num_flds, Num_Grids, no_flds, cumul, irow, jrow, prob, location, jocation, elevation, cumul_TV, irow_TV, jrow_TV)

Sub GIS_data_land(User_Loc, location, k_loop)

Reads in user supplied GIS land cover type and determines if the land is ag-capable, urban, or mountains/rock/wetlands.

Sub read_data(isum, User_Loc, ii, jj, icol, SF, location, k_loop)
This subroutine reads in the raw data for land cover type for the central 3x3 townships based upon user supplied raster based grid of 10x10 grid system for each township. Data is found in worksheet "GIS_data". This subroutine is called from the subroutine "Sub_GIS_data_land"

Sub GIS_data_elev(Elev, elevation)
This subroutine reads in the raw elevation data for the central 3x3 townships based upon user supplied raster based grid of 10x10 grid system for each township. Data is found in worksheet "GIS_data".

Sub GIS_data_pop(Pop, population, k_loop)
This subroutine reads in the raw population data for the central 3x3 townships based upon user supplied raster based grid of 10x10 grid system for each township. Data is found in worksheet "GIS_data".

Sub Pop_null()
This subroutine places zeros into each grid in the worksheet "Population" and is activated by the macro button found in this worksheet.

Sub Elev_null()
This subroutine places zeros into each grid in the worksheet "Elevation" and is activated by the macro button found in this worksheet.

Sub Land_null()
This subroutine places zeros into each grid in the worksheet "LandCover" and is activated by the macro button found in this worksheet.

Sub write_Misc3(JX, X_R, Y_R, Z_R, AREA_R, RATE_R, DEPTH_R, IDATE_R, AMASS_R, CMASS_R, D_or_S_R, ITARP_R, SINCORP_R, SYR_R, i, ic, sw_x, sw_y, sw_z, Fld_size, app_rate, m_actual, ApDate, k_loop)
Write to the ASCII file "field_repeat.dat" that contains information on location and field size for fields that will be retreated in the following year. Call crystal ball to obtain new application rates and dates, depth of incorporation etc. for these fields that will be retreated in the next year. Even though the field sizes/locations may remain the same, then can get a different application treatment as given by the variability associated with the user supplied PDFs for crop types.

Sub Reinitialize_Layout(location)
This subroutine converts the areas were fields were placed in the previous year back to ag land.

Sub update_loc_array(i, ic, rand1, rand2, N_grid, location, jocation)
This routine updates the "location" array which contains integer entries for land type. For location >0, it denotes land that can't be used (1 = water, 2 = urban, 3 = mts, 4 = previously used

T*V. For location <=0, then this land has been used for previous field types, but can also be used again if required for the random sampling methodology (wt_ID = 0). For the township section weighting (wt_ID = 1), once a field is located, then this land can't be used for future fields in the given year of simulation. This avoids having one field boundary overlapping another field, especially in regions where a section has a high probability of being treated over other township sections.

Sub Get_random_Num_fit(rand1, rand2, ic, N_grid, location, jocation)
This subroutine is called when the system has tried to place a field into the user defined sections but has failed. Failure is probably due to the section and neighboring sections being full. So, randomly place this field somewhere within the township that has ag-capable land. Note: This subroutine should probably never be called unless the user has made some input errors such as a large township allocation and small numbers of sections in a township where the fields may go. Default fld_used = 0. If field can't fit, then the argument brought back from this routine is fld_used=1

Sub GIS_data_for_LandCover()
This subroutine takes GIS information from the worksheet "GIS_data" and summarizes it in the worksheet "LandCover" such that a landcover graphic can be plotted up. In this way, the user can check out the land cover and the temporal changes in land cover (if forecast_ID = 1) before a simulation is initiated.

Sub GIS_data_for_Population()
This subroutine takes GIS information from the worksheet "GIS_data" and summarizes it in the worksheet "Population" such that a landcover graphic can be plotted up. In this way, the user can check out the land cover and the temporal changes in land cover (if forecast_ID = 1) before a simulation is initiated.

Sub Get_Fld_Coordinates(ic, iteration, N_Count, sw_x, sw_y, sw_z, Twn_External_ID, Xcoord_SW, Ycoord_SW, repeat_now, fld_used, wt_ID, i, prob_neighbor, neighbor, irow_neighbor, jrow_neighbor, cumul_neighbor, Xbound, Ybound, section_TV, prob_TV, i_section_TV, irow_TV, jrow_TV, cumul_TV, rand1, rand2, section, prob, i_section, irow, jrow, cumul, N_grid, location, jocation, Fortran_opt, jfill, no_flds, elevation, kp, kcount, jflag, jcount, k_repeat, X_REP, Y_REP, Z_REP)
This subroutine determines the Southwest corner of field coordinates (sw_x, sw_y, sw_z). Elevation data for the z-coordinate is avalable for the townships within the central 3x3 but is not available for outside townships. Thus, a constant elevtion is assumed for all townships that are outside the central township of interest. This constant elevation is the average elevation for the central 3x3 township domain. Twn_External_ID is the ID for the external townships (i.e., for the 23x23). So the ID can vary from 1 to 529

Sub Get_Coord_Sec_Wt_Ext(Xcoord_SW, Ycoord_SW, sw_x, sw_y, Sec_wt)
This subroutine has section weights for a township stored in the array Sec_wt(i), i = 1 to 36 (since this is the maximum number of sections per twnship). (Xcoord_SW ,Ycoord_SW) = Southwest corner of the township in question (meters)

Sub LandCover_color(location, jocation, LANDflag, k_loop)
This subroutine reads in the user supplied information from the worksheet "LandCover" and stores it in the array "Location(ID,i,j)". Since this array is an argument in the subroutine, one can call this routine to bring in the array "Location(ID,i,j)" into other subroutines.

Sub Pop_plot_Update(k, istart, iend, User_Pop, MaxPop, MinPop, Num_incr, dinc)
This routine updates graphics found in sheet "Population" for simple & crude contour plots (no smoothing) for elevation and population information that is contained in numeric for in the worksheet "GIS_DATA".

Appendix C. SOFEA Subroutine Connectivity Diagrams

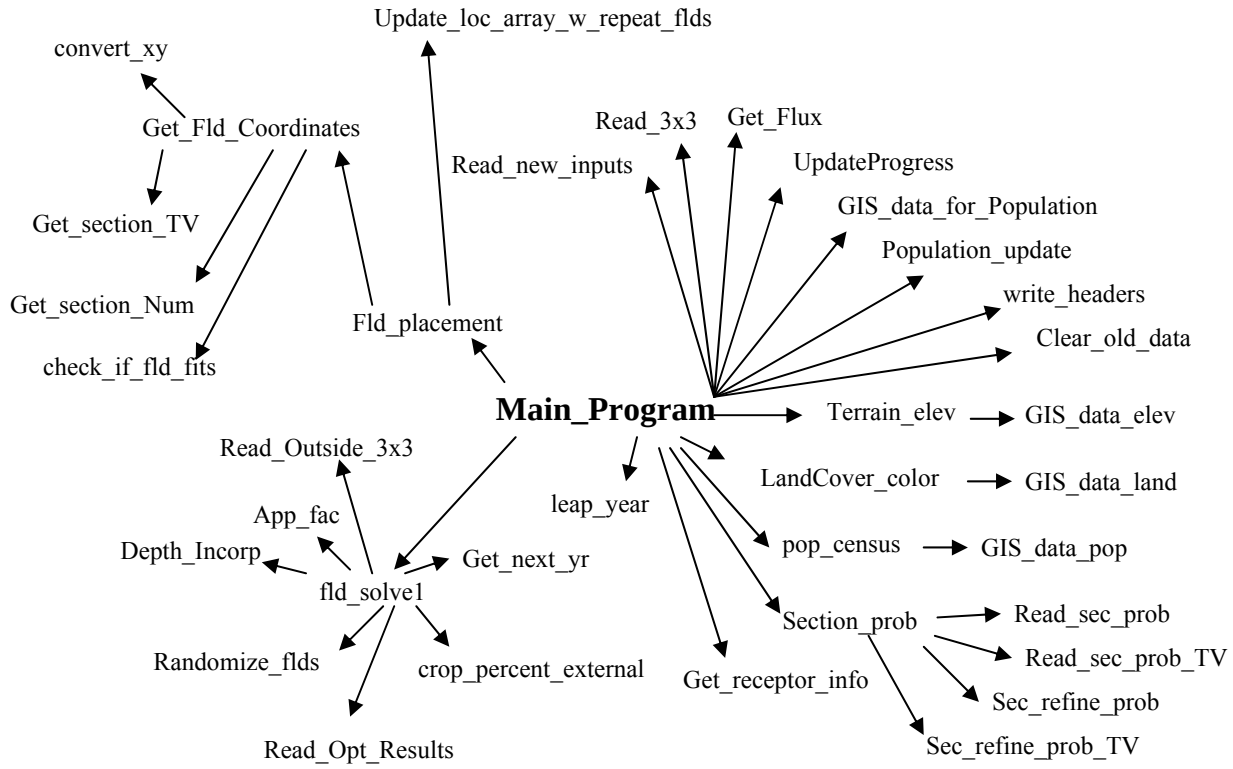

Figure D.1 Connectivity of subroutines to Main_Program

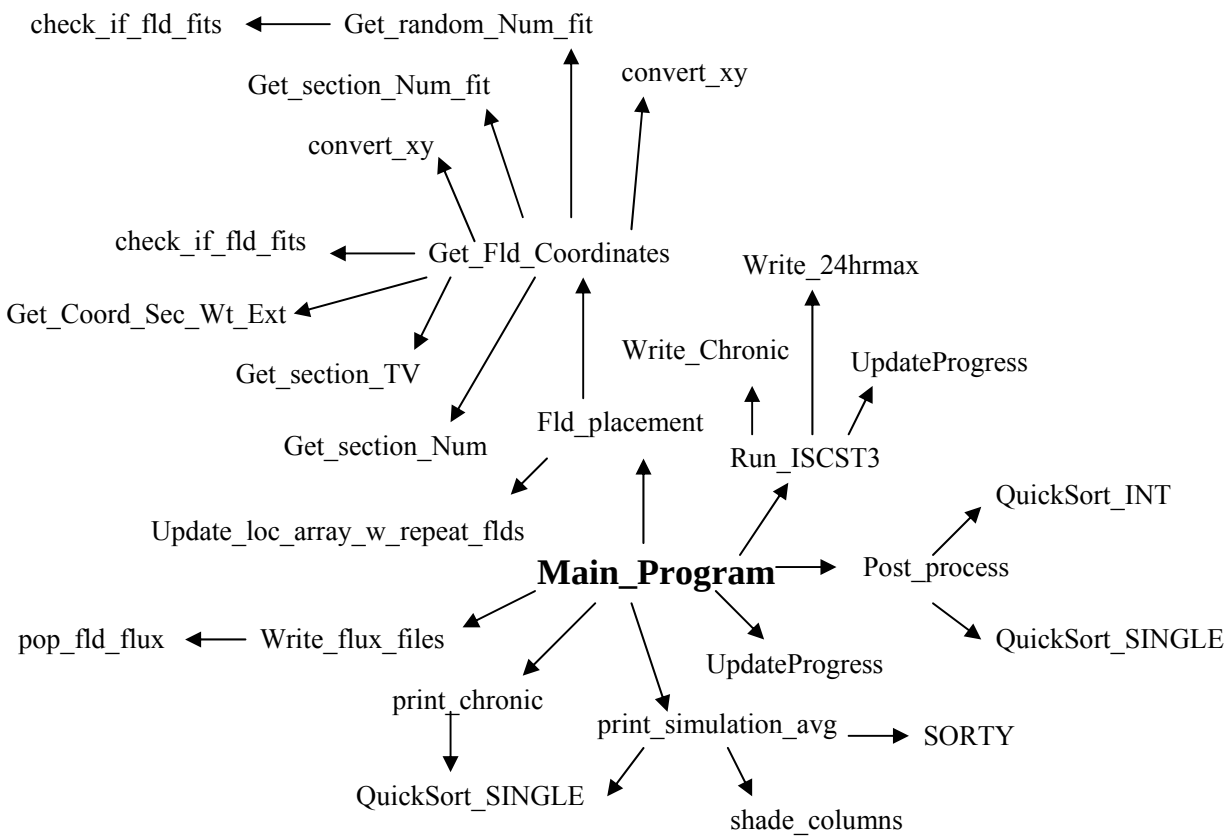

Figure D.2 Connectivity of subroutines to Main_Program (Cont'd).

Appendix D. Names and Definitions of Parameter used in SOFEA

Parameter	Description
A_End(ic, j)	Ending field size after optimization for township ID “ic”, and source ID “j”
A_Start	Starting field size before optimization (i.e., that sampled from the appropriate PDF) for township ID “ic”, and source ID “j”
Act_Mass(k, l)	PDF sampled (determined from PDF values) “actual” mass for source term for crop ID “k” and Field source ID “l”. Actual mass is the field size * application rate.
AEND	Variable used to store ending field size information when reading in the optimization program generated file “Field_opt.dat”
Am_i_drip	Random number between 0-1. Used in subroutine “fld_solve1” to determine if the application is a drip or shank.
anemo_ht	Anemometer height where meteorological wind speed/direction data was gathered. User specified in worksheet "PDF Parameters"
Ap_type(i,j)	Integer flag describing drip (1) or shank (2) applications for crop "i" and application # "j" for crop "i", where j 1, isrc(i)
ApDate	Julian application date for current field
ApDepth(i)	Application (soil incorporation) depth for crop “i” (cm)
APP	Variable used to store Julian application date information when reading in the optimization program generated file “Field_opt.dat”
App_date(i,j)	Application Date (Julian) for crop type "i", tonwship ID "TID", and max number of crop treated fields in simulation year for crop "i" (i.e., itot(ic,i)), App_date(i, TID, itot(ic,i))
app_drip_ref	Worksheets("Flux_files").Cells(4, 4)
App_rate	Application rate [kg/ha]
app_shank_ref	Worksheets("Flux_files").Cells(4, 1)
App_type(I,j,k)	Denotes the type of application. App_type = 0, then we have a drip application, for all other integers we have a shank application. I = crop ID, j = Township ID, k = Source ID
Appdate(i, j)	Application date for township ID “i” and source ID “j”
Area(k, l)	PDF sampled field area for crop ID “k” and Field source ID “l”
Area_end	ending field size after optimization
Area_start	Starting field size before optimization (i.e., that sampled from the appropriate PDF
AST	Variable used to store starting field size information when reading in the optimization program generated file “Field_opt.dat”
ave	Holder variable to read in value from ISCST3 output that isn’t required or used
avg_per(np)	User specified running average periods (i.e., 3-day, 60-day, etc.). np 1 to Num_avg_per
Boat_load_memory	A user supplied flag to tell program if post-processing routines are to be used. Much of the Post-processing entails storing large amounts of data in memory until the simulations are complete, after running averages etc. can be determined. If this parameter equals 1, then detailed 24-hr ISCST3 output files are generated (Gb) and the subchronic post processing routine is executed. If 0, then no subchronic information is generated.
bound	A bound on the integer number of treated fields for current year of simulation that will include repeat fields (used in subroutine “fld_solve1“)
Bufzone	Buffer zone setback around treated fields [m], read in from worksheet "PDF Parameters"
c1	Variable used in subroutine “Post_Process” to store and write concentration information to output summary worksheets.
C2	Variable used in subroutine “Post_Process” to store and write concentration information to output summary worksheets.

CA_13D_Scen	If = 0, then SOFEA uses the methodlogy for adjusting the actual mass using the Application Factors of CDPR that are specific for 1,3-D only (Thus, CA_13D_Scen = 0 for 1,3-D only, and only if 1,3-D is used in CA). If a region other than CA is simululated, then =1. This parameter = 1 for all soil fumigants, and will only = 0 for 1,3-D in the state of CA only.
chron_conc(i)	Same information as in Chronic_all(j,i), except this array is used for a single year of simulation. When simulation year is complete, values in this array are stored in the appropriate location in Chronic_all(j,i).
Chronic_all(j,i)	Chronic exposure (annual) where all crop types have been superimposed. J = 1 to the total number of years simulated, and I = 1 to the total number of receptors in simulation domain.
Chunk	Increment for percentile output (i.e., print out every "ichunk" value) to reduce the size of data in percentile function. ichunk is based upon a user supplied input "Num_percen_pts". Used to generate parameter "ichunk"
cmax	Maximum concentration observed during a single simulation year
cmax1	Largest running average receptor concentration over entire year for crop #1 (TV). Used to populate the array "conc_avg_fld_max1(i)"
cmax2	Largest running average receptor concentration over entire year for crop #2 (FC). Used to populate the array "conc_avg_fld_max2(i)"
cmax3	Largest running average receptor concentration over entire year for crop #3 (NC). Used to populate the array "conc_avg_fld_max3(i)"
cmax4	Largest running average receptor concentration over entire year for crop #4 (SB). Used to populate the array "conc_avg_fld_max4(i)"
cmax5	Largest running average receptor concentration over entire year for crop #5 (PP). Used to populate the array "conc_avg_fld_max5(i)"
CNO	Character string "no" for output fields (i.e, is a tarp present?...)
CO_LINE(I)	ISCST3 Key word input file strings for COntrol pathway
col_num	Integer value used when specifying which column to write to in summary output worksheets (a.k.a. "Chronic")
col_num1	Integers used to specify what column to write summary information to in output worksheets such as "Chronic"
col_num2	Integers used to specify what column to write summary information to in output worksheets such as "Chronic"
conc(i,j,k)	24 hr average concentration that is read in from the ISCST3 output file summarizing this information. Here, i crop id (1 TV, 2 FC, 3 NC, 4 SB, 5 PP), j integer number of 32,500 dimension increments in the 24hr concentration array, k unique receptor ID within the 32,500 dimension. For example, lets assume the total number of 24hr/receptor concentrations within the ISCST3 24hr file is 50,000. Then, for TV crops, the concentration information is stored in "conc(1,j,k)", where j 1,2 and k 1, 32500 when j 1, and k 1, 50,000-32500 when j 2. In this way we can overcome the dimension and integer constraints implemented in the version of Excel this model was developed in.
conc_avg_fld_max1()	Maximum running average air concentration for crop ID #1 (TV) observed during year
conc_avg_fld_max2()	Maximum running average air concentration for crop ID #2 (FC) observed during year
conc_avg_fld_max3()	Maximum running average air concentration for crop ID #3 (NC) observed during year
conc_avg_fld_max4()	Maximum running average air concentration for crop ID #4 (SB) observed during year
conc_avg_fld_max5()	Maximum running average air concentration for crop ID #5 (PP) observed during year
conc_avg_fld1()	Running average air concentration for crop ID #1 (TV), conc1/IDAY
conc_avg_fld2()	Running average air concentration for crop ID #2 (FC) , conc2/IDAY
conc_avg_fld3()	Running average air concentration for crop ID #3 (NC) , conc3/IDAY
conc_avg_fld4()	Running average air concentration for crop ID #4 (SB) , conc4/IDAY
conc_avg_fld5()	Running average air concentration for crop ID #5 (PP) , conc5/IDAY
conc_avg_twn(i, j)	Running average concentration for receptor "i" and day of simulation year "j"

conc_avg_twn_max(i)	Maximum running average concentration for receptor “i” over entire year of simulatoin
conc_fld1(i,j)	Array where 24-hr information from ISCST3 yearly simulation fir crop #1 (TV) is stored. Here “i” is the receptor ID and j = day of year.
conc_fld2(i,j)	Array where 24-hr information from ISCST3 yearly simulation fir crop #2 (FC) is stored. Here “i” is the receptor ID and j = day of year.
conc_fld3(i,j)	Array where 24-hr information from ISCST3 yearly simulation fir crop #3 (NC) is stored. Here “i” is the receptor ID and j = day of year.
conc_fld4(i,j)	Array where 24-hr information from ISCST3 yearly simulation fir crop #4 (SB) is stored. Here “i” is the receptor ID and j = day of year.
conc_fld5(i,j)	Array where 24-hr information from ISCST3 yearly simulation fir crop #5 (PP) is stored. Here “i” is the receptor ID and j = day of year.
conc_twn(jk, IDAY)	The receptor concentration from all field types (i.e., the sum of the concentrations from all simulations at a particular location and day. The location is via receptor # "jk", and the day is "IDAY". Note: conc_twn(jk, IDAY) is the 24-hr concentration of a unique receptor on a given day that is the sum of all individual crop type simulations.
conc1	Sum of air concentration for crop ID #1 (TV)
conc2	Sum of air concentration for crop ID #2 (FC)
conc3	Sum of air concentration for crop ID #3 (NC)
conc4	Sum of air concentration for crop ID #4 (SB)
conc5	Sum of air concentration for crop ID #5 (PP)
Concentration(i)	Chronic concentration (annual average) for receptor “i”
Corr_Mass(k, l)	PDF sampled (determined from PDF values) “corrected” mass for source term for crop ID “k” and Field source ID “l”. Corrected mass = actual mass * SF, where SF is the scaling factor determined by CDPR methodology
cputime	Actual cpu time for simulation to complete.
crop_hit(j)	Crop ID number for crop field types having 1,3-D treated fields.
crop_hit_id(j)	Flag used to determine if a specific crop type has any source terms (0 = no, 1 = yes). If no, then the model doesn’t execute ISCST3 for this crop type “j”.
csum_neighbor	Sum of probabilities for all neighboring sections that boarder a non-zero probability user supplied section.
Cum_flux	Cumulative flux total for given time of year. If reference flux profile was obtained in the summer, the Cum_Flux Pct_vol_ref. However, if the reference flux profile was obtained in the cool season, then the summer time loss would be “Cum_Flux” equals Pct_vol_ref*1.6.
cumul(1 To 9, 1 To 36)	Cumulative distribution array for user supplied township section probabilities for Township ID “I”, j = 1 to i_section(ic) for annual crops
cumul_neighbor(1 To 9, 1 To 100)	Cumulative distribution array for neighboring sections to non-zero supplied township section probabilities for Township ID “I”, j = 1 to i_section(ic) for annual crops (used by spillover algorithm)
cumul_neighbor_TV(1 To 9, 1 To 100)	Cumulative distribution array for neighboring sections to non-zero supplied township section probabilities for Township ID “I”, j = 1 to i_section(ic) for perennial (TV) crops (used by spillover algorithm)
cumul_TV(1 To 9, 1 To 36)	Cumulative distribution array for user supplied township section probabilities for Township ID “I”, j = 1 to i_section(ic) for perennial crops (TV)
cumulative(1 To 36)	Cumulative distribution for user specified section weights as a function of townships.
CYES	Character string “yes” for output fields (i.e, is a tarp present?...)
d(1 To 12)	Number of days in each month for non-leap year
D_or_S(k, l)	PDF sampled integer flag that denotes if a field application is for drip or shank for crop ID “k” and Field source ID “l”
Date_one(1 To 9, 1 To 5, 1 To 300)	First year of information (in multiple year simulation) for application date stored in this array for township ID “I”, crop ID “j”, and source ID “k”
Date_type(i,j)	Application date (Julian) for crop "i" and application # "j" for crop "i", where j 1, isrc(i)

DeltaEle	Elevation change increment used in generating crude contour plot found in worksheet "elevation".
depth(k, l)	PDF sampled application soil depth for crop ID "k" and Field source ID "l"
Depth_ref	Depth of incorporation for reference drip flux
Depth_incorp	Depth of incorporation
DL(1 To 12)	Number of days in each month for leap year
Drip/Shank	If application is drip (0) or shank (1)
E_Type(i)	Elevation of grid "i" [m]
Elev(30, 30)	Coarse elevation data read from user supplied worksheet for 3x3 township domain (each township is a 10 x 10 grid).
Elevation	Elevation information array "elevation(twn ID, i,j)", where i and j 1, 100 (row and column).
elevation(i, j, k)	Elevation data for township I, row j and column k.
ELEVflag	Flag to tell program if GIS data for land cover exists in the worksheet GIS_Data (0 no, 1 yes)
Ext_twn_cap(i)	Fraction of the township allocation for all townships receiving source terms, where "i" = 1 to the total number of townships receiving sources (N_ext_twn_Hits)
Ext_Twn_ID(i)	Array that stores township ID's for all townships in the 23x23 domain that receive source terms. Township ID (integer from 1 to 529, see worksheet "Twn_Mass_Wt_Ext"), with 1 the top left corner township, and 529 the bottom right township. Have a total of 529 townships, with the 3x3 simulation domain at the center of this "23x23" township area. i = 1 to the total number of townships having source terms.
f_repeat	Flag used to mark a field for retreatement the following year
F_scle_Type(i,j)	Flux scale factor used to multiply the applcation rate scaled reference flux by
filename	Filename (String) for ISCST3 output file that VBA will open up and subsequently read in the simulation results.
fld_nxt(k)	If a total of "tot_fields" are required for a given township, then the array "fld_nxt(k)" will have "tot_fields" entries. Any T&V entry is given a 0. For example, if tot_fields = 10 and there are 3 T&V fields then the array fld_nxt(k) will contain three zeros, and the numbers 4, 5, 6, 7, 8, 9, and 10 randomly placed in the array for non-T&V fields. Now, if the user has selected an input that 50% of the fields (non T&V) will be retreated the following year, then any field where fld_nxt(k) >= 3 + INT(0.5*7 + .5) (i.e., 7) will receive an integer flag ID of 1 that will indicate the field will be reused the following year.
fld_nxt_yr(ic,k)	Integer flag if the field is to be used in the next year of simulation. If 0 (no), =1, yes. Thus, for random placement, fld_nxt_yr(k) 0 for all k. For section placement, this array is populated based upon the user supplied number for the percent of fields retreated in the following year (per_nxt_yr) and of course omitting any T&V fields from the retreat analysis. (note: new T&V locations can be obtained in subsequent years such that the percent of various crop types per township is met via the optimization stuff. "ic" township ID, and k 1 to total number of field in township "ic"
Fld_one(1 To 9, 1 To 5, 1 To 300)	First year of information (in multiple year simulation) for field size stored in this array for township ID "I", crop ID "j", and source ID "k"
FLD_R(i, j)	Flag used to mark if a field will be retreated in the subsequent year of simulation for township ID "i" and source ID "j". = 0 no, = 1 yes. Note: if = 1 then we already have (x,y,z) coordinates of field (X_REP, Y_REP, Z_REP)
fld_repeat(ic,i)	Flag for if this field is to be repeated in the next year of simulation (0 = no, 1 = yes). ic = township ID (1-9), i = source term ID (1 to # of sources in the year)
Fld_size(i,j)	Field size [ha] for crop "i", i=1,5 and for township ID, 1-9
fld_storage(i)	Fields ID numbers from "lowerbound" to "upper bound" are randomized and stored stored in the array fld_storage(i), where "i'" ranges from lowerbound to upperbound. This array used to select fields for retreatment the following year (i.e., if 50% retreatment specified, use 1st half on entries in array).
Flux_drip(i)	Hourly flux data that is used for drip applications (i.e., raw data can be in chunks of 6

	hr or 12 hr. Hourly values for a 6 hour chunk are assigned the observed 6-hr average). Flux is scaled by reference drip study application rate.
Flux_drip_raw(i)	Observed flux data (flux) for reference drip field study
Flux_shank(i)	Hourly flux data that is used for shank applications (i.e., raw data can be in chunks of 6 hr or 12 hr. Hourly values for a 6 hour chunk are assigned the observed 6-hr average). Flux is scaled by reference shank study application rate.
Flux_Shank_raw(i)	Observed flux data (flux) for reference shank field study
forecast_ID	Integer flag to tell program if different input parameter "loops" are to be simulated. A loop is defined as a time interval (i.e., 5 years) where specific PDFs are to be sampled, etc. The next "loop" will sample a whole new set of PDFs in such a way to "forecast" anticipated agronomic practices and resulting effects on air concentrations.
forecast_loop	Number of years of simulation per forecast "loop". User specified in worksheet PDF_Parameters.
FR	Variable used to store repeat field integer (0 or 1) information when reading in the optimization program generated file "Field_opt.dat"
Ftype(1 To 5)	Character string for crop type (TV, FC, NC, SB, PP)
func_left	This is the fraction (approximate) of total simulation left for the iterations (95%). Used when plotting dialog box with % complete.
func_per_yr	This is the fraction of total for each year of iteration
func_per_yr_cropID	This is fraction of total for each crop type for each year of simulation. We include additional time for non-ISCST3 simulation stuff
GIS_ID	Flag to tell program if GIS data is user input and found in the worksheet "LandCover" or is obtained from GIS databases and is found in the worksheet "GIS_Data"
Grid_space	Receptor spacing of uniform grid of receptors [m]
GridX	x-location [m]within township [m] defined and used in the subroutine "Get_receptor_info"
GridY	y-location [m] within township [m] defined and used in the subroutine "Get_receptor_info"
hgrid(j)	Elevation for receptor j
HGT	Elevation measurement used in subroutine "Get_receptor_info" (not currently used)
hr_init_drip	The military hour of the day the reference field study drip application was made
hr_init_shank	The military hour of the day the reference field study shank application was made
I_already_have	Integer flag used in logic found in various subroutine to denote of all source terms have been accounted for.
i_drip	Total number of hourly flux measurements/predictions in the Flux_Drip array (integer)
I_External	Flag that is used to determine if sources are placed in the central 3x3 domain only (=0) or both the central 3x3 and surrounding townships (23x23 simulation domain (=1)
i_flg	Integer denoting crop type; =1 Tree and Vine (T&V), 2, Field crops (FC), 3, Nursery crops (NC), 4, Strawberries (SB), 5, Post Plant Vine
I_ran_or_sec	User defined flag for source placement outside the central 3x3 to specify section weighting (1) or random weighting (0)
i_section(ic)	Number of sections within township "ic" having non-zero probabilities (i.e., fields can be place here.
i_section_TV(1 To 9)	Number of sections within a given township having non-zero probabilities for perennial (TV) field placement for the central 3x3 township domain.
i_shank	Total number of hourly flux measurements/predictions in the Flux_Shank array (integer)
i_test	Integer used to determine if current year is a leap year
I_twn	Integer flag to tell subroutine "fld_solve1" that we will find source terms for the central 3x3 domain (I_twn = 0). If I_twn = 1, then fld_solve1 will find source terms for a single township that is outside the 3x3 central domain.
Ibs	Counter for writing rows of information when troubleshooting
ic	Township ID (1 thru 9) for central 3x3

Icent	Integer used in leap year subroutine to denote millennium (1900 or 2000)
ICHECK_LOOP	Error trapping integer for the total number of forecast Loops. Currently, only up to 5 unique loops can be specified.
Ichunk	Integer denoting how many data points to exclude when truncating an exceedence curve data (i.e., Ichunk = 10, then every 10^{th} point from the distribution will be selected and written to a summary output worksheet). Increment for percentile output (i.e., print out every "ichunk" value) is to reduce the size of data in percentile function. ichunk is based upon a user supplied input "Num_percen_pts" and is the integer approximation for the parameter "chunk"
Icolor	Integer used to select colors used in crude contour plot found in worksheet "elevation".
icount	Integer counter used for row number for writing to output worksheets such as "24hr_Summary"
icyr	4 digit simulation year (Same as IYRO)
ID_drip_or_shank	Flag for application type (=1 for drip, 2 for shank)
ID_OPT	
IDAY	Julian day of year for output storage (1-365 or 366)
iend	Integer used to denote starting value for 3x3 township information (=30), since each township is 10x10
Ileap	Integer flag for leap year (1 non-leap year, 2 leap year)
Imax	Integer used to dynamically allocate User_Pop, User_Ele … arrays based upon the total number of "loops" (i.e., the user specifies new data for each loop in the worksheets "Population", "Elevation", and "LandCover".
incr_fine	Number of raster grids per side for a township
irow(ic, j)	Row number (1 to 6) of section "j" within township "ic" having user defined non-zero probabilities, where j = 1 to i_section(ic) for annual crops
irow_neigh(ic,i)	Row number in township "ic" for sections bordering section having non-zero prob.
irow_neighbor(ic,j)	Row number (1 to 6) of neighboring section "j" within township "ic" surrounding a user defined, non-zero section; where j = 1 to neighbor(ic). Annual Crops
irow_neighbor_TV(ic,j)	Row number (1 to 6) of neighboring section "j" within township "ic" surrounding a user defined, non-zero section; where j = 1 to neighbor(ic). Perennial (TV) Crop.
irow_TV(ic, j)	Row number (1 to 6) of section "j" within township "ic" having user defined non-zero probabilities, where j = 1 to i_section(ic) for perennial (TV) crops
iskip	Integer used to determine how many data points in overall exceedence distribution to "skip" such that total number of data points the user has specified for exceedence data is met. (Similar to parameter " ichunk")
iskip0	iskip = iskip0
isrc(k)	Number of source terms in current year of simulation for crop type "k"
Istart	Integer used to denote starting value for 3x3 township information (=1)
Itarp	Integer flag to denote of a tarp is present (=1 for no tarp, =2 for tarp)
iteration	Current integer year in total number of years to simulate loop, i.e., year 7 of 10 ..
itot(ic,l)	Maximum number of specific crop treated fields in simulation yr for field type "l", and township "ic", where 1, 5 for T&V - PP, respectively. If 2 applications are made for T&V, 4 for FC, 1 for NC, 3 for SB, and 2 for PP, then itot 4. When using information from this array, we begin at the top and select application dates down through the array. For the above example, there weill be 4 application date(s) for NC, but we will only require the first entry.
ITT	Total number of townships in the simulation domain that can/do receive source terms. ITT includes the central 3x3 (i.e. 9) plus any external townships in the 23x23 township domain that have non-zero township mass fractions.
IX(ic, k)	Number of hits (source terms) in township ID "ic", and for crop type "k"
iyr_hit	Integer counter used for column number for writing to output worksheets such as "24hr_Summary"
IYRO	4-digit year of simulation (i.e. 1988)

J_tot_Flds	Internal parameter used to dimension various arrays for the total number of source terms in the entire simulation domain (up to 23x23 townships).
jcap	Integer value used to sample appropriate PDFs for the source parameters to be selected from (for 9 townships). Currently 400.
JDAY	Integer used in post processing that ranges from 1 to the largest running average sub-chronic interval specified by user. If a 60 day running average value was selected for output, then JDAY = 1 to 60.
jocation(i, j, k)	Denotes if land has been used by another field sometime during the current simulation year. Used to keep field from being placed on top of one another in a given year of simulation
Jover	Integer used when reading in section weights for townships outside the central 3x3 (read in from worksheet "Twn_Mass_Wt_Ext")
jp_m(i,j)	Identical to the parameter “num_flds” (i.e., the total number of fields for a specific crop type), except that information is now represented by township ID “I” and crop ID “j”
Jr	Random number between 1 and jcap. This is because we have jcap entries per field for the central 3x3 stuff (i.e. 9 townships). Since we have only 1 township we are finding sources for, don't need a large list, so we sub-sample from the original 500 to something smaller.
jrow(ic,i)	Column number in township "ic" for section having non-zero probability here, i 1 to i_section(ic)
jrow_neigh(ic,i)	Column number in township "ic" for sections bordering section having non-zero prob.
jrow_neighbor(ic,i)	Column number in township "ic" for section having non-zero probability here, i 1 to i_section(ic)
jrow_neighbor_TV(1 To 9, 1 To 100)	Column number (1 to 6) of neighboring section "j" within township "ic" surrounding a user defined, non-zero section; where j = 1 to neighbor(ic). Perennial (TV) Crop.
jrow_TV(1 To 9, 1 To 36)	Column number (1 to 6) of section "j" within township "ic" having user defined non-zero probabilities, where j = 1 to i_section(ic) for perennial (TV) crops
Jul_Date(k, l)	PDF sampled application date for crop ID “k” and Field source ID “l”
Jul_ref	Julian day when the reference trial was initiated (to determine if we are in warm or cold season)
jyr	Last 2 digit integer for the year (i.e., 1998 = 98)
Jyr	2-digit year of simulation (i.e. 88)
k_loop	Current “loop” number the simulation is on.
k_repeat	Flag for any calls in this current subroutine execution to the random field placement algorithm. If we have already written out the repeat flds for the current year of execution, then k_repeat 1 and this logic will be used in the subroutine "fld_placement_random" such that the repeat fields will not be rewritten out again.
Kextend	The maximum number of days post-year that we can have when performing running averages
kfields	Total number of different crop/field types (currently 5)
kline	Integer used to write to various locations in output worksheets such as “Field_Info1”. Since Excel only allows up to 256 columns, once this constraint is exceeded, kline is some new large number (a.k.a. replace 1 with 250) such that data is now “wrapped around” much like a word processor.
kolumn_max	Error trapping integer value to see if the total number of columns for output summary exceeds Excel limitations of 256, and thus the program will need to “wrap” results around to new rows in Excel.
kover	Integer used to denote which column number in summary worksheets that information will be written to.
Kpp	Counter for error checking (tyically not used)
krepeat	Parameter used in the data results “wrapping” in worksheet "Field_Info1".
ksave	Integer flag used to save simulation results. Currently, we save after each 10-year simulation interval
Kwrap	Flag for worksheet "24hr_Summary" to wrap around when number of Excel columns

	are exceeded.
L_orig	Integer for use in reading in/determining source information for fields that may be outside of the 3x3 domain. With L_orig = 1, we are determining fields inside the 3x3 domain. For L_orig > 1 then we are determining fields outside of the 3x3 domain.
LANDflag	Flag to tell program if GIS data for land cover exists in the worksheet GIS_Data (0 no, 1 yes)
lc	Integer counter used for writing information to various rows in worksheet "Misc"
leftover	Integer that is used in the subroutine “Odd_or_Even” to determine if an integer is odd or even.
linebs	Character string used to skip header lines in ISCST3 output files
location(i, j, k)	Land cover data storage for township I, row j and column k. Array location(ID,i,j) that contains info for ag/non-ag land, and for the crop type. ID is the township ID (1-9) and i=1,100, j=1,100 (i.e. the 100x100 fine grid used for discretizing the township.
loop_cnt	Total number of annual crop fields
Lower	Integer counter used to track overall lower bound for number of fields (excluding TV) that a current year of simulation will have.
Lowerbound	Number of T&V fields plus one
M(1 To 12)	Last day of month (i.e., ML(1) 31, … ML(12) 365) for determining Julian day given calendar month and day for a non-leap year
Max_No_Twnships	Total number of townships external to the central 3x3 that have source terms.
Maxper	Randomly generated percentage for comparing with “Per_tarp” to see if the current application has a tarp or not. If a tarp is present, then the total maximum cumulative mass loss is different than for a bare soil field. Also, parameters for estimating the non-linear scaling factor with incorporation depth are also different between the tarp/un-tarped fields.
ME_LINE(I)	ISCST3 Key word input file strings for MEteorological pathway
ML(1 To 12)	Last day of month (i.e., ML(1) 31, … ML(12) 365) for determining Julian day given calendar month and day for a leap year
N_Count	Incremental integer counter for the number of fields
N_drip	Number of data points for reference drip flux profile '
N_ext_twn_Hits	
N_ext_twn_Hits(5)	Total number of townships external to the central 3x3 that will receive treated fields (i.e. the township mass fraction specified by the user in worksheet “Twn_mass_Wt_Ext” is non-zero.
N_FLD_EXTERNAL(ii)	Total number of source terms for all crop types for township “ii” external to central 3x3
N_grid	Number of raster grids of size “space” for each length of a treated field
N_loop	Number of fields for township IC and crop J
N_loops	Total number of loops (intervals when parameters can change) over the entire simulation interval.
N_shank	Number of data points for reference shank flux profile
N_yrs	User specified number of years to simulate
NCOL	Integer used to determine the number of columns required when writing data to the worksheet "Field_Info1". If NCOL > 256, then results are wrapped around.
neighbor(ic)	The total of all surrounding sections to a given non-zero user probability section annual crops). This integer is used for integer indexing up to "neighbor(ic)", where ic is the township ID.
neighbor_TV(1 To 9)	The total of all surrounding sections to a given non-zero user probability section (perennial (TV) crop). This integer is used for integer indexing up to "neighbor(ic)", where ic is the township ID.
nhit_crp	Integer flag to denote if a certain crop type ISCST3 simulation had occurred. ISCST3 simulations WON'T occur if no fields were selected for a specific crop type (i.e., the % area of this crop type was zero)
NITER	Same as parameter “iteration”.

no_flds(ic, j, k)	Flag used to denote if ag-land can support the field that is trying to be placed (0=yes, 1=no). ic = township #,j = fld type, k = Iteration year
Nspace	Number of grids per side of township (Integer). User specified in worksheet "PDF Parameters".
NTV_fields(i)	Number of TV fields in township "i", where i = 1 to 529 (i.e., the entire 23x23 township domain).
num	Integer of upper bound minus lower bound
Num_avg_per	Total number of averaging periods for subchronic exposure intervals
num_entries	Integers used to specify row to write summary information to in output worksheets such as "Run_avg_twn".
num_flds	Number of fields of particular Fld_Size(i) for crop "i", i=1,5. This is an integer
Num_Grids()	Number of grids (refined grid size) per field side for crop "i", i=1,5. This is an Integer.
Num_percen_pts	User specified number of points written for exceedence profile data. Usually, a few additional points are added to capture the extreme upper percentiles of the distribution (i.e., 99.9, 99.95, 100)
num_srcs(j)	Number of fields in current year of simulation for crop ID j, where j 1, 5
num_srcs_all(I, j)	Total number of soure terms (fields) in year "i" of simulation for crop ID j, where j = 1, 5
NYR_loop	Number of unique "loops" over simulation interval. Each loop can have different PDF values to create the ability for forecasting.
O_D_or_S(i, j)	Flag for a drip or shank application for township ID "i" and source ID "j" (=1 drip, =2 shank)
OD	Variable used to store depth of incorporation information when reading in the optimization program generated file "Field_opt.dat"
Odepth(ic, j)	Depth of incorporation for source term in township ID "ic", and source ID "j"
ODS	Variable used to store flag denoting if a drip or shank application was made when reading in the optimization program generated file "Field_opt.dat"
Oitarp(i, j)	Flag for if a tarp is present for township ID "i" and source ID "j" (=1 yes, =2 no)
OT	Variable used to store flag denoting if a tarp was present when reading in the optimization program generated file "Field_opt.dat"
P_type(i)	Population density for grid "i"
P_Type(i)	Population of grid that receptor is in
Pct_vol_max	Percent of application volatilized if applied to the surface of the soil (user supplied)
Pct_vol_ref	Percent of application that volatilized during the reference flux trial
PctDone	Percent of overall simulation complete (for updating progress bar)
per_drip	Percent of applications that are drip
per_nxt_yr	User supplied percentage of fields that are retreated on a yearly basis
Per_tarp	Percentage of applications that have a tarp over the soil surface
perc_area(ic, k)	User supplied crop percentages for townhip "ic" (ic = 1 to 9) and crop ID "k" (k = 1 to 5)
Percent_Drip(i)	Percent of applications that are drip (vs. Shank) for each crop type "(i)". This value is obtained from sampling the appropriate PDF functions given in worksheet "PDF Parameters"
percentile(j)	Percentile array that is written to output summary worksheets. Percentile = (receptor ID ["j"])/(total number of receptors)*100, where j = 1 to total number of receptors
period_max	The maximum number of days post-year that we can have when performing running averages
Pop(30, 30)	Coarse population density data read from user supplied worksheet for 3x3 township domain (each township is a 10 x 10 grid).
POPflag	Flag to tell program if GIS data for land cover exists in the worksheet GIS_Data (0 no, 1 yes))
population(i, j,k)	Population density information data storage for township I, row j and column k.
prob(ic,i)	Non-zero probability of sections in township "ic", where i 1 to i_section (36 since this

	is the maximum number of sections per township)
prob_neighbor(ic, neighbor(ic))	Probability of a "neighboring" section receiving a treated field once the user defined non-zero probability section becomes "filled". Note that this probability the original user defined probability for the central section divided by the total number of neighboring sections within township "ic".
prob_TV(i,j)	Storage location for user supplied township probabilities for perennial (TV) field placement, I = township ID (1-9), j = section ID (1-36). Note: this array is only updated if the section has a non-zero probability.
pst_name	Pesticide/pollutant name used as the POLLUTID parameter in the ISCST3 input file (i.e., this character string is used in the comments in ISCST3 output files.
r_test	Variable used to determine if current year is a leap year
R_type(i)	Land cover type for grid i [0 ag land, 1 water, 2 urban, 3 mountains]
rand1	Random integer between 1 and 100
rand2	Random integer between 1 and 100
random1	Random number between 0-1
random2	Random x location within a township section
random3	Random y location within a township section
Rate(k, l)	PDF sampled application rate for crop ID “k” and Field source ID “l”
rate_coeff	1st order decay coefficient in air
rate_k	Exponential decay rate constant used when scaling flux loses with application depth when nonlinear scaling is specified.
Rate_one(I, j, k)	First year of information (in multiple year simulation) for application rate stored in this array for township ID “I”, crop ID “j”, and source ID “k”
Rate_type(i,j)	Application rate (kg/ha) for crop "i" and application # "j" for crop "i", where j 1, isrc(i)
rdivid	Used in the subroutine “Odd_or_Even” to determine if an integer is odd or even. rdivid is defined as integer being brought into the subroutine divided by 2
RE_LINE(I)	ISCST3 Key word input file strings for REceptory pathway
rec_ht	Receptor height (m). User specified in worksheet "PDF Parameters".
recep_all	Flag specified by user to tell program if receptors are to be placed in all of the central 3x3 (1) or only the single, central 1x1 township (0)
REDIS	Character string used when writing ISCST3 input file
region	Character string for name of region weather came from (i.e., “California”)
remainder	Used in the subroutine “Odd_or_Even” to determine if an integer is odd or even. remainder = rdivid - Int(rdivid)
Repeat_flds(i)	Total the number of possible repeat fields for the following simulation year (but don't include T&V fields) for the central 3x3 for township ID “i”
repeat_now	Flag of field is a repeat from previous year (if repeat_now 1, then we already have (x,y,z) coordinates of field since this is a repeat field from the previous year of simulation
Round	Round up (1) or Round down (0) grid for field placement (User specified Celll B80 in worksheet “PDF_Parameters”.
Scal. Factor	CDPR factor for time of year and application depth (This is the # used for correcting 1,3-D mass
Scale_flg	Scale factor flag for emission flux which is a function of incorporation depth (linear = 1, non-linear = 2). User supplied parameter in worksheet “PDF_Parameters”.
SCF	Variable used to store CDMS Application Factor when reading in the optimization program generated file “Field_opt.dat”
Sdir	Active workbook path (character string)
sec_no_bordering	Count for the total number of sections not boarding a user defined section having a probability of having treated fields
Sec_wt(i)	User specified section weights (probabilities) for a specific township section outside the central 3x3 receiving treated fields (read in from worksheet "Twn_Mass_Wt_Ext")
section(i, j, k)	Stores township section weighting for annual field placement if section weighting was specified. Data is stored in array Section(Twn ID, 1 to 6, 1 to 6). Thus, each

	township is broken into 36 sections as given by the row, column integers in the array.
section_TV(i, j, k)	Stores township section weighting for perenial (TV) field placement if section weighting was specified. Data is stored in array Section(Twn ID, 1 to 6, 1 to 6). Thus, each township is broken into 36 sections as given by the row, column integers in the array.
SEED_ID	Random seed integer specified by user (currently not functional as intended)
SF_incorp(i, j)	Dimensionless incorporation depth scaling factor for flux loss (Sincorp) for township ID "i" and source ID "j"
SF_yr(i, j)	Dimensionless temporal scaling factor for flux loss (Syr) for township ID "i" and source ID "j"
Sincorp	Flux scaling factor for depth of incorporation used to scale experimental flux observations which were performed at some prescribed depth.
Slp	Slope used in linear scaling with incorporation depth.
SO_LINE(I)	ISCST3 Key word input file strings for SOurce pathway
space	Dimension of raster grid (Cell E16 of worksheet "PDF_Parameters")
Starttime	Clock time when simulation is initiated
sur_air_ID	Integer ID for the surface meteorological station where met file was measured
sw_x	x-location for northwest corner of field edge
sw_x_one(i, j, k)	Array for first year of information (in multiple year simulation) for x location of southwest corner of field for township ID "I", crop ID "j", and source ID "k"
sw_y	y-location for northwest corner of field edge
sw_y_one(i, j, k)	Array for first year of information (in multiple year simulation) for y location of southwest corner of field for township ID "I", crop ID "j", and source ID "k"
sw_z	z-location (elevation) for northwest corner of field edge
sw_z_one(i, j, k)	Array for first year of information (in multiple year simulation) for z location (elevation) of southwest corner of field for township ID "I", crop ID "j", and source ID "k"
Syr	Flux scaling factor for time of year
Tarp	If bare field (0) or tarp is present (1)
Tarp_expt	Integer flag which specifies if a tarp were used in the reference field trial experiments [0 no tarp, 1 tarp]
Tarp_YN(i, j)	Array version of the parameter "itarp" for crop ID "i" and field/source ID "j"
Tcap_frac	Fraction of the township allocation for townships external to the central 3x3 as input in the worksheet "Twn_Mass_Wt_Ext"
terrain	Character string for ISCST3 simulations (either "FLAT" or "ELEV" if elevation is specified
time_bufzone	Time [days] post application when the buffer zone constraint is active. Read in from worksheet "PDF Parameters"
Time_drip(i)	Observed time (hours post application) when flux data was measured for reference drip field study
Time_Shank(i)	Observed time (hours post application) when flux data was measured for reference shank field study
tot_fields(i)	Total number of source terms in the central 3x3 for township ID "i"
tot_receptors	Total number of receptors in simulation domain
tot_repeat	the actual number of fields that will be retreated in the following year. Excludes T&V.
town_allyear(i, j)	Array that stores the "N_yrs" average receptor concentration for each receptor in the simulation domain. Here, I = averaging period ID (1 = first period defined by user, 3 = 2nd …), and j = receptor ID
town_avg(i,j,k)	Running average receptor concentration (i.e., 1-day, 15-day, etc.), where i user specified running avg. period (1 to Num_avg_per), j receptor number (1 to kc2), and k simulation year (1 to N_yrs). This populated array is passed into this subroutine. Data in this array is obtained from reading the 24-hr data files generated by ISCST3 (array is populated in the subroutine "Post_Process"
town_avg_all()	1-D slice of 3-D array "town_avg" for running average receptor concentration for each

	receptor “j” (i.e., j receptor number (1 to kc2). Results by crop type are written to worksheet “Run_avg_twn”
town_sum(j)	"N" year average receptor concentration (based upon all field types involved) used in sorting routine for exceedence curve determination.
TwID	Internal variable for township ID, where the ID can range from 1 to the total number of townships receiving sources (N_ext_twn_Hits)
twn_1_9	Integer flag denoting if source terms for a single township external to the central 3x3 are required (=1) or if source parameters within the central 3x3 (= 9)
Twn_BS	Dummy storage variable used when reading in the first term from the SOFEA generated file "field_opt.dat".
twn_cap_base	Township allocation mass supplied by user in worksheet “PDF_Parameters”.
Twn_ID(i)	Township ID for central 3x3, i=1 to 9
Twn_Mass_Wt(i)	Township mass weight function for townships internal to the 3x3 domain, and i = 1, 9
TwnID	Integer used for tracking external townships (1 – 529, where 241, 242, 243, 264, 265, 287, 288, 289 correspond to the central 3x3 townships.
up_air_ID	Upper air Data Met Station ID
Upperbound	Total number of treated fields
User_Ele(ii, jj)	User supplied elevation information on a 10x10 grid system per township from user entries found in worksheet "Elevation" for central 3x3. Thus, this array has dimensions of 30x30.
User_Loc(ii,jj)	User supplied land cover information on a 10x10 grid system per township from user entries found in worksheet "LandCover" for central 3x3. Thus, this array has dimensions of 30x30.
User_Pop(ii, jj)	User supplied population information on a 10x10 grid system per township from user entries found in worksheet "Population" for central 3x3. Thus, this array has dimensions of 30x30.
w_file_name	Weather File Name (Character string)
wt_ID	Flag for field placement , 0 for random weighting, =1 for section weighting. This variable is a declared PUBLIC and thus the magnitude is known without coming through the subroutine argument list.
X_rate(i, j)	Application rate for township ID “i” and source ID “j”
X_REP(i, j)	x-location of SW corner of field for township ID “i” and source ID “j”.that will be retreated the following year.
x1	Dummy variable for x location [m] of a receptor read in from ISCST3 output file. We already have this information stored in other arrays.
xgrid(j)	X location for receptor j
XR	Variable used to store sw corner of field x-location information when reading in the optimization program generated file “Field_opt.dat”
XRATE	Variable used to store application rate information when reading in the optimization program generated file “Field_opt.dat”
Y_REP(i, j)	y-location of SW corner of field for township ID “i” and source ID “j”.that will be retreated the following year.
y1	Dummy variable for y location [m] of a receptor read in from ISCST3 output file. We already have this information stored in other arrays.
ygrid(j)	Y location for receptor j
YR	Variable used to store sw corner of field y-location information when reading in the optimization program generated file “Field_opt.dat”
Z_REP(i, j)	z-location (elevation) of SW corner of field for township ID “i” and source ID “j”.that will be retreated the following year.
z1	Dummy variable for elevation [m] of a receptor read in from ISCST3 output file. We already have this information stored in other arrays.
ZR	Variable used to store sw corner of field z-location (elevation) information when reading in the optimization program generated file “Field_opt.dat”