Appendix A: Reported Program Bugs

by Jy-Pyng Sah

There are eight versions of PADRE have been installed between September 13, 1995 and December 15, 1995. Table A-1 summarizes these versions and their basic characteristics.

Version 1: Installed September 13, 1995

The following problems with this version were reported:

- The save and open functions in file menu do not work properly in this version. It cannot open previously saved scenario files.
- There is a possible error in unit conversion. All the parameters for unit conversion in this program are correct except one, the number of yards in a mile. The program shows that there are 5,280 yards in a mile. Actually, the conversion parameter is exactly the number of feet in a mile rather than yards in a mile. The outputs of this calculation seem to be correct.
- The health effect levels in PADRE are not the same as ORNL-6615. For example, the 50% lethal rate (GB, adult male, light activity) in PADRE is 35 mg-min./m^3 while it was 70 mg-min./m^3 in ORNL-6615.
- Mixed time budget has been used in PADRE. That is, unadjusted time budget data for locations and adjusted time budget data for overriding media were used to estimate warning efficiency. Thus it makes the sum of time budget probabilities across all activities and locations is greater than one at certain times of the day. The use of this mixed time budget data creates a slight overestimation of warning efficiency during these periods.
- In multiple-run mode, if both protection measures are evacuation or respiratory protection and if the plume has not passed the at-risk area by the end of the simulation (3 hours), the ADR is "0". This answer is not consistent to single-run's ADR. However, if at least one protection measure is in-place shelter, the program shows the correct ADR in multiple-run and marks that the plume has not passed yet.

Version 2: Installed October 30, 1995, with the previous save-open bugs corrected.

The following problems with this version were reported:

- It shows inconsistent value for behavioral factor in scenario and warning system windows. Opening any saved scenario file after saving another scenario, results in an unchanged behavioral parameter in scenario window. Nevertheless, the behavioral parameter on warning system window correctly shows new value. This change of parameter seems to effect the dosage and probability curve results.
- The parameter for 30-minute limit also revealed the inconsistent value problem. In addition, certain values for 30-minute limit of the warning

system window showed a 1 percent decrease in the Scenario windows (e.g., 60 to 59, 94 to 93).

- Showing the warning system window led the social and technological diffusion factor to 0, and eventually changes the model results. Switching among preset warning systems led to the same result. Basically, the diffusion factors are not "0" for any of the "canned" systems. The specified system parameters from ORNL 6615 are not logically reflected in these systems.
- In addition, during the running of this program, it crashed occasionally due to the HIBYTE and LOBYTE errors in the WINDUTIL.CPP.

Version 3: Installed November 9, 1995

The following problems with this version were reported:

- This version of PADRE failed to run due to insufficient memory in our DOS compatible Macintosh. Even if it could run on 486-based IBM compatible PC, it couldn't show the graph of dosage result.

Version 4: Installed November 27, 1995

The following problems with this version were reported:

- This version of PADRE failed to run after installation. It caused an application error that the error message shows WINFILE caused a general protection error in KRNL386.EXE at 0002:17A5.

Version 5: Installed December 1, 1995, had the warning system problems fixed.

The following problems with this version were reported:

- None of the speed-keys in this version function properly (e.g., Ctrl-R doesn't run the program and Ctrl-E doesn't show the scenario window).

Version 6: Installed December 4, 1995, had part of the speed-keys installed.

The following problems with this version were reported:

- This version has most of its speed-keys installed, but Alt-F4 does not function correctly to close a window. It shows no immediate effect after Alt-F4 been pressed. However, after several other key-strokes or commands, it shows effect that stopped the PADRE.

Version 7: Installed December 8, 1995, had most of the speed-keys installed.

The following problems with this version were reported:

- In default scenario, if the down-wind distance is between 11,993 and 12,114 meters, the program crashes when run.

Version 8: Installed December 15, 1995, has health effects levels corrected and downwind distance problem in previous version corrected.

The following problems with this version were reported:

- Several speed-key options are not working (e.g., Alt-h for both help and enhanced in the shelter in place window, and Alt-b for both before and basic sirens of the warning system window).
- The 30-minute limit problems remains. It shows error only when the 30-minute limit is 58 (%) in warning system window, the scenario window shows 57 (%).

Table A-1: Summary of Reported Bugs

Date	Size (in bytes)	Comment(s)
1. September 13, 1995	413,696	• Cannot opened previously saved scenario; • Possible error in unit conversion; • Inconsistent LCt50 and LCt1 in PADRE and ORNL-66 • Time budget leads to overestimation; • Multiple-run error.
2. October 30, 1995	373,248	• Inconsistent value in scenario window and warning system window; • Warning system I/O error in opening dialogs (values going to 0); • Preset diffusion factors in warning system lost; • Application program errors.
3. November 9, 1995	418,304	• Application error.
4. November 27, 1995	366,898	• Application error.
5. December 1, 1995	366,080	• No speed-keys.
6. December 4, 1995	367,104	• Short of some speed-keys (i.e., Alt-F4).
7. December 8, 1995	294,176	• Crashed in certain down-wind distance range.
8. December 15, 1995	333,680	• Short of some speed keys; • 30-minute limit problem remains.

Appendix B: Code Verification Commentary

by Steven Eichner

The member of the investigation team responsible for the code verification process did not have access to ORNL 6615 until after he determined the output of the various procedures through code analysis. He did have access to both the code and working version of PADRE. The isolation from 6615 was important because it allowed an unbiased evaluation of code results and produced a successful comparison of the PADRE code with the theoretical background contained in the earlier work.

Evacuation Protection Capacity

(Calc Evac protection capacity)

This procedure is located in the Main section and is responsible for calculating the protection capacity for evacuation. It accomplishes this by multiplying the unprotected exposure by 1 minus the joint probability of completing the evacuation. This value is calculated for each minute between 0 and *gNumber of Minutes* , which is set equal to 181, the total number of minutes.

The engine used is:

```
protecion_cap[min] = (1.0 - *((double *) gImplementHndl + min)) * (*((double *)
* gConcentrationHndl + min)).
```

It returns an array called protection_cap, the same name used regardless of protection method (e.g. evacuation, in-place protection, etc.).

The code used to achieve this is:

```
static void calc_evac_protection_cap(double *protection_cap)
{
        short min;

        MoveHHi(gPImplementHndl);
        HLock(gPImplementHndl);

        for (min = 0; min < gNumberOfMinutes; min++)
        {
                protection_cap[min] = (1.0 - * ((double *) *gPImplementHndl + min)) * (*((double *)
*gConcentrationHndl +  min));
        }
        HUnlock(gPImplementHndl);
}       /* calc_evac_protection_cap() */.
```

Calculation of In-place Protection Capacity

(Calc_In_Place_Protection)

This procedure calculates the protection capacity for in-place protection and returns the results in an array called protection_cap. Each element of this array contains the sum of the previous minute's protection capacity plus the concentration factor plus the number of minutes minus 1 minus the previous minute's protection capacity times the rate of air exchange.

The engine used is:

protection_cap[min] = protection_cap[min - 1] + (rate_air_exchange * (*((double*) *gConcentrationHndl + min -1) - protection_cap[min - 1]));

The code used to achieve this is:

```
static void calc_in_place_protection_cap(double *protection_cap)
{
        short min;
        double rate_air_exchange = gAirChangesPerHour / 60.0;

        /* CONCENTRATION LOCKED BY THE CALLING PROCEDURE */
        protection_cap[0] = rate_air_exchange * (*((double *) *gConcentrationHndl + 0));

        for (min = 1; min < gNumberOfMinutes; min++)
        {
                protection_cap[min] = protection_cap[min - 1] + (rate_air_exchange * (*((double *)
*gConcentrationHndl + min - 1) - protection_cap[min - 1]));
        }
}       /* calc_in_place_protection_cap() */.
```

Calculation of Respiratory Protection Capacity

Calc_Resp_Protection_Cap

This procedure calculates the protection capacity for respiratory protection. The cumulative unprotected dosage is calculated by summing the concentration incremented by minute for the period between 0 and 181.

The concentration increment is set to ConcentraionHndl, BreakThrough is set to FALSE, and the cumulative unprotected dosage is set to zero prior to the index loop which is controlled by the minute.

The unprotected dosage accumulates the concentration increment and the protection capacity for that minute is given the value of that minute's concentration increment times the leakage rate until the cumulative dosage is greater than the pre-established breakthrough value as developed by the code extraction:

```
protection capacity= concentration_increment[min] * leakage
              //     b = 1.0;
              //protection_cap[min] = ((1.0 - b) * ConcentrationIncrement[min] *
gLeakageRate) + b * (ConcentrationIncrement[min]).
```

After breakthrough occurs, protection capacity for that minute is set to the difference between the breakthrough value and the accumulated unprotected dosage minus that minute's concentration increment. Added to this is the difference between the cumulated unprotected dosage and the breakthrough value, producing an equation such as:

protection_cap[min] = leakage_rate * (breakthrough - cumulative_unprotected_dosage - concentration_increment[min]) + cumulative_unprotected_dosage - breakthrough.

After the "breakthrough" minute, protection_capacity[min] is set to the concentration increment for that minute as represented by:

protection_cap[min] = concentration_increment[min].

The following table presents the selections in summary form:

Table B-1.

Prior to breakthrough	At the minute of breakthrough	After breakthrough
protection capacity= concentration_increment[min] * leakage	protection_cap[min] = leakage_rate * (breakthrough - cumulative_unprotected_dosage - concentration_increment[min]) + cumulative_unprotected_dosage - breakthrough.	protection_cap[min] = concentration_increment[min]

The code below is the segment that is responsible for this comparison.
*

```
    for (min = 0; min < gNumberOfMinutes; min++)
    {
        CumulativeUnprotectedDosage += ConcentrationIncrement[min];
        if (CumulativeUnprotectedDosage > gBreakthroughValue)
        {
            if (BreakThrough == FALSE)
            {
                BreakThrough = TRUE;
                protection_cap[min] = (gBreakthroughValue -
CumulativeUnprotectedDosage - ConcentrationIncrement[min]) * gLeakageRate;
        protection_cap[min] += CumulativeUnprotectedDosage - gBreakthroughValue;
//Calculate dosage above breakthrough
            }
            else
                protection_cap[min] = ConcentrationIncrement[min];
        }
        else
            protection_cap[min] = ConcentrationIncrement[min] * gLeakageRate;
prot += protection_cap[min];
        //      b = 1.0;
        //protection_cap[min] = ((1.0 - b) * ConcentrationIncrement[min] * gLeakageRate) +
b * (ConcentrationIncrement[min]);
    }
}
```

Calculation of Exposure Curves

(Calculate_exposure_curves)

This procedure calculates the exposure curves, calling (as appropriate) calc_evac_protection_cap, in_place_protection, and respiratory protection.

The first process the procedure invokes is to call the appropriate protection measure that establishes the correct initial array values for calc_evac_protection. The next step is a process that assigns the joint probability to a variable called resAsjExpPtr[min] using the formula:

```
resAdjExpPtr[min] = (jointp[min2] * protCapPtr[min]) + ((1.0 - jointp[min2]) *
        (ConcentrationHndl + min)).
```

Following this step, the computer proceeds to calculate the total expected exposure, which is equal to the integral of exposure from time 0 to minute *t* using the code segment:

```
for (min = 0; min < gNumberOfMinutes; min++)
 //     {
    //      for (min2 = 0; min2 <= min; min2++)
    //          totalExpPtr[min] += resAdjExpPtr[min2];
//}.
```

The next step involves the creation of the y-components for the exposure curves. Again, this is done through using a loop structure based upon the number of minutes (181). At the start of each pass through the loop, the cumulative capacity is increased by the protected capacity, the cumulative adjusted respiratory protection is increased by the adjusted exposure, and the cumulative exposure is increased by the concentration plus the minute.

If in-place protection was selected and the difference between the protected capacity and the concentration plus minute is greater than 0, the cumulative capacity for vacating is increased by the concentration plus minute as is the cumulative respiratory adjusted vacated value (ResAdjVac). If the difference is equal to or less than 0, the cumulative capacity for vacating is increased by the protected capacity and the cumulative adjusted respiratory actions are increased by the adjusted exposure.

```
static void calculate_exposure_curves(double *jointp)
{
    plotPointer pdata;
    //BOOLEAN plume_arrived;
    short min;
    short min2;
    short curve;
    double *protCapPtr,/* Protection Potential Increment*/
    resAdjExpPtr,  /* Population Average Increments*/
    //*totalExpPtr, /* Unprotected Increments        */
    cumultvCap = 0.0, /* Cumulative Protection Potential*/
    cumultvCapvac = 0.0, /* Cumulative Protection Potential (Vacated)*/
    cumultvResAdj = 0.0, /* Cumulative Populatino Average (i.e. adjusted for response) */
    cumultvResAdjvac = 0.0, /* Cumulative Protection Potential (Vacated)                    */
    cumultvExp = 0.0, /* Cumulative Unprotected Dosage                                      */
        max_y_val = 0.0;
        Handle protCapHndl, resAdjExpHndl;// totalExpHndl;

        MoveHHi((Handle)  gExpHandle);
        HLock((Handle) gExpHandle);

        pdata = *gExpHandle;
        /* Set the plot title and axis labels      */
        ((char *) pdata->title, "Dosage & Concentration Curves");
        strcpy((char *) pdata->legendX, "Minutes From Start of Agent Release");
        strcpy((char *) pdata->legendY, "mg-min/cu m");
        /* Set the X-Axis parameters   */
        pdata->xlo = 0.0;                          /* Plot starts at min = 0           */
        pdata->xhi = (double) gNumberOfMinutes;    /* Plot ends at the last minute */

        strcpy((char *) pdata->Curves[0].legend, "Unprotected");
        pdata->Curves[0].line_type = CURVE; /* Unprotected                          */
        pdata->Curves[1].line_type = CURVE; /* Protection potential          */
        pdata->Curves[2].line_type = CURVE; /* Population Average            */
        /* Allocate memory for the data arrays*/
        protCapHndl = NewHandleClear((long) ((gNumberOfMinutes) *sizeof(double)));
        MoveHHi(protCapHndl);
        resAdjExpHndl = NewHandleClear((long) ((gNumberOfMinutes) *sizeof(double)));
        MoveHHi(resAdjExpHndl);
        //totalExpHndl = NewHandleClear((long) ((gNumberOfMinutes) *sizeof(double)));
        //MoveHHi(totalExpHndl)

        protCapPtr = ((double *) *(protCapHndl));               /* dereference handle data pointers
*/
        resAdjExpPtr = ((double *) *(resAdjExpHndl));
        //totalExpPtr = ((double *) *(totalExpHndl));  /* Set the appropriate protection
capacity*/
        if (gProtectionType == EVACUATION)
                calc_evac_protection_cap(protCapPtr);
        else if (gProtectionType == IN_PLACE)
                calc_in_place_protection_cap(protCapPtr);
        else
                calc_resp_protection_cap(protCapPtr);

        // !!!!!!!!!!!!!!! fails on file load
        for (min = 0, min2 = gMinsBeforeAccident; min < gNumberOfMinutes, min2 <
gNumberOfMinutes + gMinsBeforeAccident;
                 min++, min2++)
                resAdjExpPtr[min] = (jointp[min2] * protCapPtr[min]) + ((1.0 - jointp[min2]) * (*
                (( double *) *gConcentrationHndl + min)));
        /* Calculate the total expected exposure which is the integral of         */
        /* exposure from time zero to minute t.
        */
//      for (min = 0; min < gNumberOfMinutes; min++)
//      {
        //      for (min2 = 0; min2 <= min; min2++)
        //              totalExpPtr[min] += resAdjExpPtr[min2];
        //}
        /* Create the exposure curves                                              */

        for (min = 0; min < gNumberOfMinutes; min++)
        {
                cumultvCap += protCapPtr[min];
                cumultvResAdj += resAdjExpPtr[min];
                cumultvExp += * ((double *) *gConcentrationHndl + min);

//              if ((*((double *) *gConcentrationHndl + min) > IGNORABLE) &&
(gProtectionType == IN_PLACE))
                if (gProtectionType == IN_PLACE)
                {
                        //plume_arrived = TRUE;
                        if ( (protCapPtr[min] - *((double *) *gConcentrationHndl + min)) >
ZEROVALUE)
                        {
                                cumultvCapvac += * ((double *) *gConcentrationHndl + min);
                                cumultvResAdjvac += * ((double *) *gConcentrationHndl + min);
                        }
                        else
                        {
                                cumultvCapvac += protCapPtr[min];
                                cumultvResAdjvac += resAdjExpPtr[min];
                        }

                }
                /* Store the data into the plot curves    */
                pdata->Curves[0].vector[min] = cumultvExp;              /* Unprotected
                                                                                  */

                pdata->Curves[1].vector[min] = cumultvCap;              /* Protection potential or
Protection potential, not vacated         */
                pdata->Curves[2].vector[min] = cumultvResAdj;           /* Population average or
Population average, not vacated                       */

                if (gProtectionType == IN_PLACE)
                {
                  pdata->Curves[3].vector[min] = cumultvCapvac; /* Protection potential, vacated
*/
                  pdata->Curves[4].vector[min] = cumultvResAdjvac;/* Population average,
vacated */
                }
        }
        // for (min)


        HUnlock(gConcentrationHndl)   /*******  END LOCK CONCENTRATION *******/
        /* Set the number of curves
                        */
        if (gProtectionType == IN_PLACE)
                pdata->n_curves = 5;
        else
                pdata->n_curves = 3;

        pdata->n_points = gNumberOfMinutes;  /* temporarily always 181 points for exposure
curve */

         /* Create the LCT curves.  If the time of acident is in the middle          */
        strcpy((char *) pdata->Curves[2].legend, "Population average");
        strcpy((char *) pdata->Curves[1].legend, "Protection potential");

        // dlp4 if (gTimeOfAccident > 5)
        // dlp4{
        switch (gAgentType)    /* Mild activity levels */
        {
                case GB:
                        pdata->EPD = 0.33;              /* Evacuation/Protective Distance */
                        pdata->LCT1 = 10;               /* One percent lethality         */
                        pdata->LCT50 = 35;              /* Fifty percent lethality       */
                        break;
                case VX:
                        pdata->EPD = 0.25;              /* Evacuation/Protective Distance */
                        pdata->LCT1 = 4.3;              /* One percent lethality         */
                        pdata->LCT50 = 15;              /* Fifty percent lethality       */
                        break;
                default:                                // This is for HD
                        pdata->EPD = 2;                                                 /*
Evacuation/Protective Distance */
                        pdata->LCT1 = 150;              /* One percent lethality        */
                        pdata->LCT50 = 900;             /* Fifty percent lethality      */
                        break;
        }

        //      strcpy((char *) pdata->Curves[pdata->n_curves].legend, "ECt50 (miosis) resting
newborns");
        pdata->Curves[pdata->n_curves].line_type = LCT;

        /* set the Y-Axis parameters                                                    */
        for (curve = 0; curve < pdata->n_curves; curve++)
        {
                for (min = 0; min < pdata->n_points; min++)
                        max_y_val = mMAX(pdata->Curves[curve].vector[min], max_y_val);
        }
```

Warning System

The warning system (human factor) variables are set in a code segment that the developers themselves suggest should be revisited. Rather than gathering and calculating the information in completely seperate procedures, there is significant mixture of code that will make it difficult to modify in the future. One portion of this code is the routine save_system_parameters, which, while its main purpose is to verify system parameters, also places the appropriate user-defined values into k, a1, a2, and limit.

```
static BOOLEAN save_system_params(DialogPtr dialogPtr)
{
        double dtemp;
        short itemp;

        if (!get_double_TE_verify(dialogPtr, I_Technological, "Technological", &dtemp,
DIFF_k_LWR, DIFF_UPR, 0, NULL, ""))
                return FALSE;
        current.UserSetRow.k = dtemp;

        if (!get_double_TE_verify(dialogPtr, I_Behavioral, "Behavioral", &dtemp,
DIFF_a1_LWR, DIFF_UPR, 0, NULL, ""))
                return FALSE;
        //current.UserSetRow.a1 = dtemp;
        DisplayA1_2 = dtemp;
```

```
                        if (!get_double_TE_verify(dialogPtr, I_Social, "Social", &dtemp,
0.0, DIFF_UPR, 0, NULL, ""))
                              return FALSE;
                        current.UserSetRow.a2 = dtemp;

                        if (!get_short_TE_verify(dialogPtr, I_30_Min_Limit, "30 Minute
Limit", &itemp, 1, 100, 0, NULL, ""))
                              return FALSE;
                        current.UserSetRow.limit = (double) itemp / 100.0;

                        return true;
}
```

The procedure DoOKWarningSystem saves the data from the Warning System dialog for use elsewhere in PADRE. Like the procedure above, it is a boolean function which returns true when the dialog has been sucessfully completed by the user. The main points of interest for the immediate purpose of allocating values for parameters may be found in the code segment:

```
gWarningSystem[gW_User_Set_Row].a2 = current.UserSetRow.a2;
gWarningSystem[gW_User_Set_Row].limit = current.UserSetRow.limit;
gWarningSystemType = current.WarningSystemType;
switch (gWarningState)
{
      case SET_TIME:
            if (!save_min_warn_complete(dialogPtr))
                  return FALSE;
            current.WarnUserSetComplete = true;
            break;
      case CHOOSE_SYSTEM:
      case SET_DIFFUSION:
            if (!save_system_params(dialogPtr))
                  return FALSE;
            current.WarnUserSetComplete = false;
            break;
}
gWarningSystem[gW_User_Set_Row].k = current.UserSetRow.k;
if (OriginalA1_2 != DisplayA1_2)
{
      gWarningSystem[gW_User_Set_Row].a1 = DisplayA1_2;
      gParametersEverDirty = 1;
}
gWarningSystem[gW_User_Set_Row].a2 = current.UserSetRow.a2;
gWarningSystem[gW_User_Set_Row].limit = current.UserSetRow.limit;
```

As can be noted above, gWarningSystem[gW_User_Set_Row].a2 = current.UserSetRow.a2; and gWarningSystem[gW_User_Set_Row].limit = current.UserSetRow.limit; appear twice. One occurence of each should be removed.[1]

[1] ORNL comments: An astute observation! We have removed the redundant instructions.

The most important code segment calculates the rate for release. This is accomplished in load_warning_system, which uses k, a1, and limit to construct the percentage warned by using the functions limit *=rate and p_warned += k * (gWSA1 * (limit - p_warned)) + (1.0 - k) * (a2 * p_warned * (limit - p_warned)). These formulae are calculated throughout the entire modeling period (181 minutes) and are represented by the following code sequence:

```
        gWSA1 = GetA1(gWarningSystemType);
        rate = gWarningSystem[gW_User_Set_Row].rate = 1.0 + (1.0 -
gWarningSystem[gW_User_Set_Row].limit) / 100.0;

        HLock(gPWarningHndl);

        *((double *) *gPWarningHndl) = 0.0;

        for (min = 1; min < MAX_MINS; min++)
        {
                limit *= rate;
                p_warned += k * (gWSA1 * (limit - p_warned)) + (1.0 - k) * (a2 * p_warned *
                (limit - p_warned));
                *((double *) *gPWarningHndl + min) = mMIN(p_warned, 1.0);
        }
```

This procedure also contains the equation that produces the warned curve, using the equation: Pw += k * (a1 * (L - Pw)) + (1.0 - k) * (a2 * Pw * (L - Pw)).

The following procedure defines the DoOKWarningSystem procedure and brings to a close the presentation of the warning system.

```
static BOOLEAN DoOKWarningSystem(DialogPtr dialogPtr)
{

        // Save decision to warn variables
        short *valueptr;
        short lower_limit;
        short upper_limit;
        char *message;

        // Minutes before/after accident, read from same field, interpretted based on
time_direction.
        current.MinsAfterAccident = 0;
        current.MinsBeforeAccident = 0;
        if (time_direction == TIME_BEFORE)
        {
                valueptr = &current.MinsBeforeAccident;
                message = "Minutes Before";
                lower_limit = MIN_BFR_LWR;
```

```
            upper_limit = MIN_BFR_UPR;
    }
    else
    {
            valueptr = &current.MinsAfterAccident;
            message = "Minutes After";
            lower_limit = MIN_AFTR_LWR;
            upper_limit = MIN_AFTR_UPR;
    }
    if (!get_short_TE_verify(dialogPtr, I_Min, message, valueptr, lower_limit, upper_limit,
0, NULL, ""))
            return FALSE;

    // Verification complete.  Safe to set global variables
    gMinsBeforeAccident = 0;
    gMinsAfterAccident = 0;
    if (time_direction == TIME_BEFORE)
            gMinsBeforeAccident = *valueptr;
    else
            gMinsAfterAccident = *valueptr;
    gWarningSystem[gW_User_Set_Row].a2 = current.UserSetRow.a2;
    gWarningSystem[gW_User_Set_Row].limit = current.UserSetRow.limit;
    gWarningSystemType = current.WarningSystemType;
    switch (gWarningState)
    {
            case SET_TIME:
                    if (!save_min_warn_complete(dialogPtr))
                            return FALSE;
                    current.WarnUserSetComplete = true;
                    break;
            case CHOOSE_SYSTEM:
            case SET_DIFFUSION:
                    if (!save_system_params(dialogPtr))
                            return FALSE;
                    current.WarnUserSetComplete = false;
                    break;
    }
    gWarningSystem[gW_User_Set_Row].k = current.UserSetRow.k;
    if (OriginalA1_2 != DisplayA1_2)
    {
            gWarningSystem[gW_User_Set_Row].a1 = DisplayA1_2;
            gParametersEverDirty = 1;
    }
    gWarningSystem[gW_User_Set_Row].a2 = current.UserSetRow.a2;
    gWarningSystem[gW_User_Set_Row].limit = current.UserSetRow.limit;
    gWarnUserSetComplete = current.WarnUserSetComplete;

    // User set time to complete warning
    gMinWarningComplete = current.MinWarningComplete;

    load_decision_to_warn();
    load_warning_system();
    return TRUE;
}   // DoOKWarningSystem
```

Probability for Decision To Warn.

The decision to warn uses two separate procedures, joint_probability and load_decision_to_warn to produce the desired results. Load_decision_to_warn converts the input collected from a user dialog and converts it into a negative value (if warning occurs prior to an accident) and leaves it positive if the warning occurs after the event. In either case, the value is placed into minuteWarned, which reflects *when* the warning is issued.

The second code segment produces the warning curve activation. It accomplishes this by putting 0 into the array dtw for those minutes prior to warning and pushes 1 into all subsequent minutes through 180. The following code segment is used in the current version of PADRE:

```
if ((*gProbHandle)->minuteWarned > 0)
        {
                for (short k = 0; k < (*gProbHandle)->minuteWarned; k++)
                        dtw[k] = 0.0;
                for (k = (*gProbHandle)->minuteWarned; k < MAX_MINS; k++)
                        dtw[k] = 1.0;
        }
        else
                for (short k = 0; k < MAX_MINS; k++)
                        dtw[k] = 1.0.
```

Code segments:

```
extern void load_decision_to_warn(void)
{
        //short min = 0;
        plotPointer ptr;

        MoveHHi((Handle) gProbHandle);
        HLock((Handle) gProbHandle);

        ptr = *gProbHandle;

        // store decision to warn min in the prob plot structure
        if (gMinsBeforeAccident > 0)
        {
                gNumberOfMinutes = 181;                                          //+
gMinsBeforeAccident //
                ptr->minuteWarned = gMinsBeforeAccident * -1;
                sprintf(gDTWBuf, "Warning system activated (min):%d", -gMinsBeforeAccident);
        }
        else
        {
                gNumberOfMinutes = 181;
                ptr->minuteWarned = gMinsAfterAccident;
                sprintf(gDTWBuf, "Warning system activated (min):%d", gMinsAfterAccident);
        }
        HUnlock((Handle) gProbHandle);

        gDecisionSet = true;
}       // load_decision_to_warn()


void joint_probability(double *j3)
{
        double tp1[MAX_MINS], tp2[MAX_MINS];
        double j1[MAX_MINS], j2[MAX_MINS], dtw[MAX_MINS];
        double jp, *jpd, *pd1, *pd2;
        short t_min = gNumberOfMinutes + gMinsBeforeAccident;
        register short t, j, i;


        if ((*gProbHandle)->minuteWarned > 0)
        {
                for (short k = 0; k < (*gProbHandle)->minuteWarned; k++)
                        dtw[k] = 0.0;
                for (k = (*gProbHandle)->minuteWarned; k < MAX_MINS; k++)
                        dtw[k] = 1.0;
        }
        else
                for (short k = 0; k < MAX_MINS; k++)
                        dtw[k] = 1.0;
```

Joint Probability Information

If one of the probability distributions is already a joint probability, then it is cumulative rather than incremental in which case it has to be converted to incremental before it can be incorporated into the new joint probability calculation.

This procedure uses an interesting programming tool by using the case statement to allow the same routine to be used for calculating the curves for the decision-to-warn as well as j1 and j2. The first time through the loop, PADRE assigns pd1 the array dtw, which contains either 1's or 0's. Pd2 is assigned the value contained in WarningHndle, and jpd is assigned the array j1. The second time through the loop, the probability for public response is calculated, and the third time through the implementation factors are considered.

The initial total probability for each of two curves is then assigned the values contained within the 0 element (time 0) of pd1 and pd2 (dtw[0] and PWarningHndl[0], respectively).

Two total probabilities are then calculated, tp1[t+1] is set equal to pd[t+1]- pd1[t] and is then modified to be equal to mMax (tp1[t+1],0.0). Similarly, tp2[t+1] is set to pd2[t+1]-pd2[t] and is then modified by mMax(tp2[t+1], 0.0). In both cases, the tp*x*[t+1] might be modified to be the *difference* between the current and future value of pd[t]. However, this is not the case because of the subsequent change in tp[t+1]. The code segment is:

```
for (t = 0; t < (t_min - 1); t++)
        {
                tp1[t + 1] = pd1[t + 1] - pd1[t];
                tp1[t + 1] = mMAX(tp1[t + 1], 0.0);
                tp2[t + 1] = pd2[t + 1] - pd2[t];
                tp2[t + 1] = mMAX(tp2[t + 1], 0.0);
        }
```

The joint probability is then calculated by summing the product of tp1 and tp2 arrays. When the probability exceeds 1, the curve is truncated by jpd[t] = mMIN((jpd[t - 1] + jp), 1.0) although it does not appear to have been implemented correctly.

The final step occurs when the protection type is IN_PLACE and the shelter action is CLOSE_DOORS_AND_WINDOWS. In this case, the curve represented by j2 is modified to reflect the time budget adjustment plus 1 minus the budget adjustment times the original j2[t] as represented by j2[i] = gTimeBudgetAdjustment + (1.0 - gTimeBudgetAdjustment) * j2[i] when implementation is started. When implementation is completed, the j3 curve is modified by using the formula j3[i] = gTimeBudgetAdjustment + (1.0 - gTimeBudgetAdjustment) * j3[i].

```
void joint_probability(double *j3)
{
        double tp1[MAX_MINS], tp2[MAX_MINS];
        double j1[MAX_MINS], j2[MAX_MINS], dtw[MAX_MINS];
        double jp, *jpd, *pd1, *pd2;
        short t_min = gNumberOfMinutes + gMinsBeforeAccident;
        register short t, j, i;

        for (i = 0; i < 3; i++)
        {
                switch (i)
                {
                        case 0:
                                pd1 = dtw;
                                pd2 = (double *) *gPWarningHndl;
                                jpd = j1;
                                break;
```

```
            case 1:
                pd1 = j1;
                pd2 = (double *) *gPResponseHndl;
                jpd = j2;
                break;
            case 2:
                pd1 = j2;
                pd2 = (double *) *gPImplementHndl;
                jpd = j3;
                break;
        }
        tp1[0] = pd1[0];
        tp2[0] = pd2[0];
        for (t = 0; t < (t_min - 1); t++)
        {
            tp1[t + 1] = pd1[t + 1] - pd1[t];
            tp1[t + 1] = mMAX(tp1[t + 1], 0.0);
            tp2[t + 1] = pd2[t + 1] - pd2[t];
            tp2[t + 1] = mMAX(tp2[t + 1], 0.0);
        }
        jpd[0] = tp1[0] * tp2[0];
        for (t = 1; t < t_min; t++)
        {
            jp = 0.0;
            j = t;
            for (short min = 0; min <= t; min++)
            {
                jp += tp1[min] * tp2[j];
                j--;
            }
// If the total probability has exceeded 1.0 we truncate the probability curve. jpd[t] = mMIN((jpd[t
- 1] + jp), 1.0);
        }
    }                                               // end for i = 0; i< 3

if ((gProtectionType == IN_PLACE) && (gShelterAction == CLOSE_DOORS_AND_WINDOWS)
&& gTimeBudgetSet)
    {
        for (i = 0; i < t_min; i++)
        {
            // Implementation started
            j2[i] = gTimeBudgetAdjustment + (1.0 - gTimeBudgetAdjustment) * j2[i];

            // Implementation completed
            j3[i] = gTimeBudgetAdjustment + (1.0 - gTimeBudgetAdjustment) * j3[i];
    }   }
    plot_joint_prob(dtw, j1, j2, j3);
}   // joint_probability()
```

Public Response

The public response issues are divided into several different program areas including Load_Public_Response, Resp.CPP, Evac.cpp, inplace.cpp, and public.cpp. Things happen in different places and it is important to understand how they interact.

The Load_Public_response procedure loads the public response probability into memory and then initially assigns a 0 as the response probability to each element. A loop is then begun which extends from the minute response started through the last minute of the simulation. Public response is assigned the value 1/ (1+padre_exp ((-1*s) * (t-rs - m)) where the last term (padre_exp ((-1*s) * (t-rs - m)) returns the exponent of ((-1*s) * ((t-rs)- m)), 0, 1, or infinity. This value is multiplied by the response scale factor and the loop is repeated until the simulation is complete. See the code below for additional information.

```
static double padre_exp(double x)
{
        if (x > 0.0)
        {
                if (x >= LIMIT)
                        return(INFINITY);
                else
                        return(exp(x));
        }
        if (x < 0.0)
        {
                if (-x >= LIMIT)
                        return(0.0);
                else
                {
                        DivideByZero( exp(-x) < ZEROVALUE);
                        return(1 / exp(-x));
                }
        }
        return(1.0);
}

extern void load_public_response()
{
        register short t, i,                                                    /* time interval */
                rs = gMinResponseStarted;
        register double s = gSlope, m = gMidpoint, rsf = gResponseScaleFactor;
        sprintf(gACCBuf8, "Downwind Distance (m) = %.2f", gDownWindDistance);
        HLock(gPResponseHndl);
        s = s * rsf;
        DivideByZero(rsf < ZEROVALUE);
        m = m / rsf;
        for (i = 0; i < MAX_MINS; i++)          /* Zero out the Response probability*/
        {
                *((double *) *gPResponseHndl + i) = 0.0;
        }
        for (t = rs; t < MAX_MINS; t++)
        {
                DivideByZero((1 + padre_exp((-1.0 * s) * ((double) (t - rs) - (double) m))) <
ZEROVALUE);
                *((double *) *gPResponseHndl + t) = 1 / (1 + padre_exp((-1.0 * s) * ((double) (t - rs) -
(double) m)));
                //*((double *) *gPResponseHndl + t) *= rsf;
        }
        if (rs > 0)
        {
                sprintf(gPubBuf, "Delay Response by (min):%d", rs);
        }
        else
        {
                sprintf(gPubBuf, "Public Response");
        }
        HUnlock(gPResponseHndl);
        gResponseSet = TRUE;
}       /* load_public_response() */
```

Evacuation

If the PADRE user has selected evacuation as the method of protection, then PADRE calculates the minutes required for evacuation to a safe distance.

Load_Evacuation

```
extern void load_evacuation(void)
{
        short minute_safe, min;
        double evac_speed, seconds, safe_distance_meters;
        double MinuteSafe;
        if (gClearanceTime < 1)
                                                        /* Convert the miles-per-hour to meters-
per-second   */
        {
                evac_speed = gAverageMph * 0.44704;
                safe_distance_meters = units("mi", "m", gSafeDistance);
                /* Check to see if the initial down wind distance is further from the accident than
the safe distance */
                if (safe_distance_meters < 0.0)
                        minute_safe = 1;
                else
                {
                        DivideByZero(evac_speed < ZEROVALUE);
                        seconds = safe_distance_meters / evac_speed;
                        MinuteSafe = (short) ((seconds / 60.0) + 0.99);
                        minute_safe = (short) MinuteSafe;
                }
        }
        else
```

```
            minute_safe = gClearanceTime;
        for (min = 0; min < MAX_MINS; min++)
        {
            if (min < minute_safe)
                *((double *) *gPImplementHndl + min) = 0.0;
            else
                *((double *) *gPImplementHndl + min) = 1.0;
        }
        sprintf(gProtBuf1, "Protection & Strategy:  Evacuation, Clearance Time = %d minutes",
gClearanceTime);
        sprintf(gProtBuf2, "Clearance Time: %d (min)", minute_safe);
        gProtBuf3[0] = '\0';
        gImplementationSet = TRUE;
        gProtectionSet = TRUE;
        //gProtectionType = EVACUATION;
}       /* load_evacuation() */

#pragma argsused()
extern DLGCALLBACK do_EVA_proc(DialogPtr dialogPtr, UINT msg, WPARAM itemHit,
LPARAM lParam)
/
```

In-Place Protection

The In-place protection procedure assigns 0 to the ImplementHndl for every minute until TimeToShelter and 1 to the handle for every minute thereafter when the shelter action is User_set_in_place. If another form of in-place protection was used, then the values are changed appropriately (through the first 45 minutes). Comments in the code suggest that additional data regarding protection actions should be generated and placed into the appropriate data files.

Do_Inplace_protection

```
        if (gShelterAction == USER_SET_IN_PLACE)
        {
            for (min = 0; min < MAX_MINS; min++)
            {
                if (min < gTimeToShelter)
                    *((double *) *gPImplementHndl + min) = 0.0;
                else
                    *((double *) *gPImplementHndl + min) = 1.0;
            }
            gImplementationSet = TRUE;
            sprintf(gProtBuf1, "Protection & Strategy: Sheltering, %d minutes needed to
complete", gTimeToShelter);
            return;
        }

        // The data file only contains values for the first 45 minutes.
        // The remaining time should be initialized as well.
        HLock(gPImplementHndl);
        for (min = 0; min < MAX_MINS; min++)
                *((double *) *gPImplementHndl + min) = 1.0;

        for (min = 0; min < 45; min++)
        {
                if (gShelterAction == CLOSE_DOORS_AND_WINDOWS)
                        * (((double *) *gPImplementHndl)  + min) = Shelter[min][0];
                else
                        *(((double *) *gPImplementHndl)  + min) = Shelter[min][1];
        }
        if (gShelterAction == CLOSE_DOORS_AND_WINDOWS)
                sprintf(gProtBuf1, "Protection & Strategy:  Sheltering, Close Doors And
Windows");
        else
                sprintf(gProtBuf1, "Protection & Strategy:  Sheltering, Tape And Seal Room");

        gImplementationSet = TRUE;
        HUnlock(gPImplementHndl);
}       /* in_place_shelter() */
```

Respiratory Protection

The respiratory protection procedures establish the breakthrough values and leakage rates which are used elsewhere in PADRE.

```
 #include "Padre.h"

//////////////////////////////////////////////////////////////////////
//      File:           Respiratory_Protecti.c
//      Function:       Handle all operations for this Modeless Dialog
//      History:        6/7/90 Original by Julian Ray, U.T. Transportation Center.
//////////////////////////////////////////////////////////////////////

struct res_data
{
        protection_t Breakthrough;
        double BreakthroughValue;
        short TimeToProtect;
        double LeakageRate;
};
static struct res_data last;
static struct res_data current;

static void do_button_select(short itemHit, DialogPtr dialogPtr)
{
if ((itemHit >= I_User_set__break) && (itemHit <= I_Chem_industry)) {
                switch (itemHit)
                {
                        case I_Chem_industry:
current.Breakthrough = CHEMICAL_INDUSTRY; current.BreakthroughValue = 480000.0; break;
case I_Chem_Agent_Worker:
current.Breakthrough = CHEMICAL_AGENT_WORKERS; current.BreakthroughValue = 412000.0;
break;
            case I_NATO_Civilian:
current.Breakthrough = NATO_CIVILIAN; current.BreakthroughValue = 480000.0; break;
            case I_NATO_Military:
current.Breakthrough = NATO_MILITARY; current.BreakthroughValue = 952000.0; break;
            default:
current.Breakthrough = USER_SET_PROTECTION; break;
        }
set_ctls_val_off(dialogPtr, I_User_set__break, I_Chem_industry); set_ctl_val_on(dialogPtr,
itemHit);       // Set the Radio button if (itemHit == I_User_set__break)
        {
            ShowDItem(dialogPtr, I_Breakthrough);
SelIText(dialogPtr, I_Breakthrough, 0, 32767);
        }
        else
        {
            HideDItem(dialogPtr, I_Breakthrough);
SelIText(dialogPtr, I_Leak_Rate, 0, 32767);
        }
    }
}   //  do_button_select

static void setup_RES_dialog(DialogPtr dialogPtr)
{
    short select;
    double LeakRate;
#if defined (_MAC)
    short saveMenu, iType;
    Handle item_hndl;
#endif
SetDLGRefCon(dialogPtr, (long) RESPIRATORY_DLG); CenterDialog(dialogPtr);

// Initialize local copies of global variables current.Breakthrough = gBreakthrough;
current.BreakthroughValue = gBreakthroughValue;
switch (current.Breakthrough)               // Setup initial conditions
    {
        case CHEMICAL_INDUSTRY:
            select = I_Chem_industry;
            break;
        case CHEMICAL_AGENT_WORKERS:
            select = I_Chem_Agent_Worker;
            break;
        case NATO_CIVILIAN:
            select = I_NATO_Civilian;
            break;
        case NATO_MILITARY:
            select = I_NATO_Military;
            break;
        case USER_SET_PROTECTION:
            select = I_User_set__break;
            break;
    }
```

```
        do_button_select(select, dialogPtr);
        current.TimeToProtect = gTimeToProtect;
        current.LeakageRate = gLeakageRate;
        last = current;

// rir940208 - moved next line down after setxxx_TE_vals below: //
        do_button_select( select, dialogPtr );
#if defined (_MAC)
set_item_proc(dialogPtr, RI_20_Box, (ProcPtr) draw_box); set_item_proc(dialogPtr, RI_21_Box,
(ProcPtr) draw_box);
#endif
setlong_TE_val(dialogPtr, I_Breakthrough, current.BreakthroughValue); LeakRate =
(current.LeakageRate * 100.0);
setint_TE_val(dialogPtr, I_Leak_Rate, (short) LeakRate); if (current.Breakthrough ==
USER_SET_PROTECTION)
SellText(dialogPtr, I_Breakthrough, 0, 32767);
        else
        SellText(dialogPtr, I_Leak_Rate, 0, 32767);
setint_TE_val(dialogPtr, I_Time_To_Protect, current.TimeToProtect);

        SHOWWINDOW((WindowPtr) dialogPtr);                                          //
Open a dialog box //
}       //      setup_RES_dialog

extern DLGCALLBACK do_RES_proc(HWND dialogPtr, UINT msg, WPARAM itemHit,
LPARAM lParam)
{
        BOOLEAN ParametersDirty;

        switch (msg)
        {
                case WM_INITDIALOG:
                        setup_RES_dialog(dialogPtr);
                        return 0;                                               // so
Windows will not override our SellText in setup_RES_dialog
                case WM_COMMAND:
#if defined (_WIN )
                        if (HIWORD(lParam) == EN_UPDATE | | HIWORD(lParam) ==
EN_CHANGE)
                                return true;                                    // want to
filter these out becaue we do it when hit
#endif
                        do_button_select(itemHit, dialogPtr);
                        switch (itemHit)
                        {
                                case HELP:
#if defined( _MAC)
                                        do_help(RESPIRATORY_DLG);
#elif defined( _WIN)
                                        do_help(LoWord(iAPPLE_MENU_ABOUT + 33));
#endif
                                        break;
                                case OK:
                                        if (DoOKButton(dialogPtr))
                                        {
                                                            load_resp_prot();
                                                            load_respiratory();
                                                            gTimeBudgetSet = false;
                                                            gProtectionType = RESPIRATORY;
ParametersDirty = !EqualStructs((char *) &current, (char *) &last, sizeof(res_data));
gParametersEverDirty  |= ParametersDirty;
                                                            if
(SetReplayMode(ParametersDirty))
{
gBreakthrough = last.Breakthrough; gBreakthroughValue = last.BreakthroughValue;
gTimeToProtect = last.TimeToProtect; gLeakageRate = current.LeakageRate;
} check_status_update(); closeDialog(dialogPtr);
                                        }
                                        return true;
                                  case Cancel:
closeDialog(dialogPtr); return true;
default:
                                        break;
                              }
      }
      return false;
}     //    do_RES_proc


////////////////////////////////////////////////////////////////////
//          NAME:     LOAD_RESP_PROTECTION()
//
//          HISTORY:     July 1990     Julian Ray
//
//          PURPOSE:     Loads repiratory protection parameters.
//
//          CALLS:
////////////////////////////////////////////////////////////////////
void load_resp_prot(void)
{
      gProtectionType = RESPIRATORY;

      switch (gBreakthrough)
      {
            case CHEMICAL_INDUSTRY:
                  gBreakthroughValue = 480000.0;
                  sprintf(gProtBuf1, "Protection & Strategy: Respirators, CSEPP Civilian
Emergency Responder");
                  break;
            case CHEMICAL_AGENT_WORKERS:
                  gBreakthroughValue = 412000.0;
                  sprintf(gProtBuf1, "Protection & Strategy: Respirators, M40/C2");
                  break;
            case NATO_CIVILIAN:
                  gBreakthroughValue = 480000.0;
                  sprintf(gProtBuf1, "Protection & Strategy: Respirators, M17/M13A2");
                  break;
```

```
                case NATO_MILITARY:
                        gBreakthroughValue = 952000.0;
                        sprintf(gProtBuf1, "Protection & Strategy: Respirators, M9/M11");
                        break;
                default:
sprintf(gProtBuf1, "Protection & Strategy: Respirators rated for %0.f mg-min/cu m",
gBreakthroughValue); break;
        }
        sprintf(gProtBuf2, "Breakthrough (mg-min/m cu):%lf", gBreakthroughValue);
        sprintf(gProtBuf3, "Mask Leakage (%%): %lf", gLeakageRate * 100);

        gProtectionSet = TRUE;

}       // load_resp_prot()
```

Protected Action- Time Budgets

This procedure handles the time budget requirements and makes the appropriate adjustments for time-dependent activities.

```
/*****************************************************************
File:           Time_Budget.c
Function:       Handle all operations for this Modeless Dialog
History:        6/7/90 Original by Julian Ray, U.T. Transportation Center.
**************************************************************    */
static BOOLEAN HomeAsleep;
static BOOLEAN HomeIndoors;
static BOOLEAN NhoodOutdoors;
static BOOLEAN InTransit;
static BOOLEAN AtWork;
static BOOLEAN Television;
static BOOLEAN Radio;
static double ProportionProtected;
static double TimeBudgetAdjustment;
static BOOLEAN TimeBudgetSet;
/*****************************************************************
**              NAME:      LOAD_TIME_BUDGET()
**
**              HISTORY:    July 1990  Julian Ray
**
**              PURPOSE:    Loads time budget parameters.
**
**              CALLS:     do_TOA()
****************************************************************/  extern  void
load_time_budget(void)
{
        double adjustment[8], total = 0.0;
        static double TimeBudgets[25][7] =
        {/* a wide array of data not present in this summary */;
        // The time of accident needs to be set before we can load time time budget curves.
        //if (!gTOASet)
        //{
        //      cParamText("The TIME OF ACCIDENT must ", "be set before the ", "dosage can be
calculated",  "");
        //      gAlertResult = CautionAlert(GEN_ALT, NULL);
        //      doModalDlg(0L, INCIDENT_DLG, 0, do_ACC_proc, generic_modal_proc);
        //      return;
        //}

        adjustment[0] = TimeBudgets[gTimeOfAccident][0];
adjustment[1] = TimeBudgets[gTimeOfAccident][1];
adjustment[2] = TimeBudgets[gTimeOfAccident][2]; adjustment[3] =
TimeBudgets[gTimeOfAccident][3]; adjustment[4] = TimeBudgets[gTimeOfAccident][4];
adjustment[5] = TimeBudgets[gTimeOfAccident][6]; adjustment[6] =
TimeBudgets[gTimeOfAccident][5];

        if (gHomeAsleep)
                total += adjustment[0];
        if (gHomeIndoors)
                total += adjustment[1];
        if (gNhoodOutdoors)
                total += adjustment[2];
        if (gInTransit)
                total += adjustment[3];
        if (gAtWork)
                total += adjustment[4];
        if (gTelevision)
                total += adjustment[5];
        if (gRadio)
                total += adjustment[6];

gTimeBudgetAdjustment = gProportionProtected * total; gTimeBudgetSet = gTimeBudgetAdjustment
> 0.0 ? TRUE : FALSE;
}       /* load_time_budget() */
```

Appendix C: Procedure for Generating Behavioral Parameter (a) in PADRE

By Jea-Guy Park

New broadcast parameters, a, along with time budget by activities and locations are determined through following procedures.

First, each proportion of population by hour and activities (Table C-1) is multiplied by each value of warning system effectiveness (Table C-2), and then summed across location/activities for each warning system to achieve the time adjusted warning system effectiveness score (Table C-3). During this process, the warning effectiveness scores for television and radio activities are changed from 'n/a' to '1.0' since the warning system in PADRE requires the use of EBS/media warning system all the times. Also, warning effectiveness scores for tone-alert radios & auto-phone, and siren & tone-alert radios & auto-phone have to be assigned based on the assumption that the combined system would be at least as effective as the most effective single warning system in the combination. Warning through siren & TA radios & auto-phone for indoor activity is assumed to be more effective than other systems. The combination of three basic systems is more effective than combining any two systems. Hence, because adding the third system "add something" to the two-way combination it is assumed to be more effective (i.e. a1=0.96 rather than 0.95).

Second, the summed values for each time budget adjusted warning system are multiplied by penetration rates of each warning systems to get new broadcast parameters, a, along with time budget by activities and locations (Table C-4).

Table C-1. Adjusted Time Budget

Hour of day	Home Asleep	Indoor	Outdoor	Transit	Work	TV	Radio
0	0.879	0.040	0.002	0.006	0.020	0.028	0.005
2	0.941	0.014	0.001	0.002	0.016	0.004	0.002
4	0.907	0.040	0.001	0.005	0.018	0.002	0.007
6	0.525	0.270	0.013	0.035	0.079	0.022	0.040
8	0.127	0.418	0.041	0.054	0.255	0.042	0.049
10	0.031	0.395	0.069	0.072	0.318	0.059	0.043
12	0.043	0.366	0.071	0.080	0.308	0.080	0.041
14	0.058	0.347	0.085	0.081	0.288	0.091	0.038
16	0.039	0.456	0.063	0.097	0.173	0.120	0.039
18	0.026	0.494	0.039	0.071	0.104	0.227	0.026
20	0.083	0.407	0.017	0.042	0.078	0.342	0.016
22	0.500	0.223	0.004	0.025	0.032	0.186	0.011
24	0.879	0.040	0.002	0.006	0.020	0.028	0.005

Table C-2. Warning System Effectiveness

	Asleep	Indoor	Outdoor	Transit	Work	TV	Radio
Basic Siren	0.691	0.800	0.900	0.900	0.600	1.000	1.000
TA-Radio	0.850	0.900	0.000	0.000	0.700	1.000	1.000
Auto-Phone	0.933	0.950	0.000	0.000	0.800	1.000	1.000
EBS/Media	0.000	0.400	0.200	0.200	0.100	1.000	1.000
Siren/TA-Radio	0.900	0.900	0.900	0.900	0.700	1.000	1.000
Siren/Phone	0.933	0.950	0.900	0.900	0.800	1.000	1.000
TA-Radio/Phone	0.933	0.950	0.000	0.000	0.800	1.000	1.000
Siren/Radio/Phone	0.933	0.960	0.900	0.900	0.800	1.000	1.000

Note: 1. Warning effectiveness values for Tone-Alert radio & Telephone and Siren & Radio & Phone systems are generated with the most effective values for the system.
2. Warning effectiveness values for Siren & Radio & Phone system in the case of indoor activity is weighted 1 percent more.

Table C-3. Sum of Warning Effectiveness*Time Budget

Hour of day	Siren	Telephone	Radio	EBS Media	Siren Radio	Siren Phone	Radio Phone	Siren Radio Phone
0	0.692	0.830	0.053	0.907	0.882	0.915	0.907	0.915
2	0.680	0.830	0.014	0.910	0.880	0.913	0.910	0.913
4	0.684	0.829	0.028	0.908	0.879	0.913	0.908	0.914
6	0.731	0.806	0.187	0.871	0.876	0.915	0.871	0.917
8	0.752	0.754	0.303	0.811	0.846	0.896	0.811	0.900
10	0.757	0.707	0.320	0.761	0.835	0.888	0.761	0.892
12	0.764	0.702	0.328	0.755	0.840	0.891	0.755	0.894
14	0.769	0.692	0.330	0.743	0.844	0.892	0.743	0.896
16	0.798	0.723	0.391	0.767	0.869	0.911	0.767	0.915
18	0.828	0.793	0.483	0.830	0.893	0.929	0.830	0.934
20	0.841	0.850	0.541	0.885	0.907	0.938	0.885	0.942
22	0.766	0.845	0.295	0.901	0.896	0.927	0.901	0.929
24	0.692	0.830	0.053	0.907	0.882	0.915	0.907	0.915

Table C-4. a1*Table c

0	0.138	0.249	0.011	0.318	0.264	0.320	0.318	0.320
2	0.136	0.249	0.003	0.319	0.264	0.320	0.319	0.320
4	0.137	0.249	0.006	0.318	0.264	0.320	0.318	0.320
6	0.146	0.242	0.037	0.305	0.263	0.320	0.305	0.321
8	0.150	0.226	0.061	0.284	0.254	0.314	0.284	0.315
10	0.151	0.212	0.064	0.266	0.250	0.311	0.266	0.312
12	0.153	0.211	0.066	0.264	0.252	0.312	0.264	0.313
14	0.154	0.208	0.066	0.260	0.253	0.312	0.260	0.313
16	0.160	0.217	0.078	0.268	0.261	0.319	0.268	0.320
18	0.166	0.238	0.097	0.291	0.268	0.325	0.291	0.327
20	0.168	0.255	0.108	0.310	0.272	0.328	0.310	0.330
22	0.153	0.253	0.059	0.315	0.269	0.324	0.315	0.325
24	0.138	0.249	0.011	0.318	0.264	0.320	0.318	0.320
a1	**0.2**	**0.3**	**0.2**	**0.35**	**0.3**	**0.35**	**0.35**	**0.35**

Appendix D: Desciption of PAECE Scenarios Compared with PADRE

By Jea-Guy Park

PADRE was compared with PAECE in terms of 33 scenarios, 12 evacuation, 15 in-place shelter, and 6 respiratory scenarios. These scenarios were selected from ORNL-6615 to represent a cross section of accident and emergency response scenarios. The expected exposure reduction estimated by PAECE are compared with those of PADRE. The 33 scenarios are composed of various combinations of agent, release duration, wind speed, wind stability, and protection types. Other parameters are kept constant: start of release-11 AM, 65 percent released, 500 meter mixing layer height, warning activation after 5 minutes of release, EBS/media, siren and telephone for warning method, 25 percent better in public response than at Confluence, and 3 kilometers downwind distance. Scenario 1 through 12 are evacuation scenarios for three different agents with the probability of completing evacuations with 1-, 5-, 10- and 20-min clearance times. Among the 12 evacuation scenarios, scenario 1 to 4 are evacuation scenarios at 20-min release duration for GB class III events when 1-m/s winds and E wind stability prevail . Scenario 5 through 8 are evacuation scenarios for VX class III events with 20-min release duration when 3-m/s winds and D wind stability prevail. Scenario 9 through 12 are evacuation scenarios for GB class V events with 20-min release duration when 3-m/s winds and D wind stability prevail.

Scenarios 13 through 27 are in-place shelter scenarios for 5 different agent classes with three types of shelter implementations of close door/windows and enhanced, tape/seal and expedient, and close door/windows and pressurized shelter. Among the 15 in-place shelter scenarios, scenarios 13 to 15 are in-place shelter scenarios for GB class V events with 20-min release duration when 3-m/s wind and D wind stability prevail. Scenarios 16 to 18 are in-place shelter scenarios for VX class V events with 20-min release duration when 3-m/s wind and D wind stability prevail. Scenarios 19 to 21 are in-place shelter scenarios for GB class I events with 1-min release duration when 1-m/s wind and E wind stability prevail. Scenarios 22 to 24 are in-place shelter scenarios for VX class III events with 20-min release duration when 1-m/s wind and E wind stability prevail. Scenarios 25 to 27 are in-place shelter scenarios for HD class V events with 30-min release duration when 3-m/s wind and D wind stability prevail.

Scenarios 28 through 33 are respiratory protection scenarios for 3 different agent classes when NATO civilian and the U.S. military respirator filter masks are applied with 15 percent mask leakage rate and 3 min implementation. Among the 6 respiratory protection scenarios, 28 and 29 are respiratory scenarios for GB class V events with 20-min release duration when 3-m/s wind and D wind stability prevail. Scenarios 30 and 31 are respiratory scenarios for VX class IV events with 30-min release duration when 1-m/s wind and E wind stability prevail. Scenarios 32 and 33 are respiratory scenarios for GB class II events with 20-min release duration when 3-m/s wind and D wind stability prevail.

Appendix E: Description of Individual Parameter Scenarios and Sensitivity Analysis Results

By Jea-Guy Park and Jy-Pyng Sah

This Appendix describes scenarios used to test the relationship between parameters and results. Hence all scenarios will hold the incident, meteorological conditions and downwind distance constant. A VO4 accident was used to reflect important credible events. Only the parameter being analyzed varies from the upper limit to lower limit in a set of runs, while the others fixed at this VO4 scenario. Thirteen parameters have been analyzed in thirteen sets of runs that composed this sensitivity analysis. Figure E1 to E3 present the parameter variation and resulting exposures.

Table E-1. Parameter Variation

	Warning Activation	Exposure Dosage	Techno logical	Exposure Dosage	Behavioral	Exposure Dosage	Social	Exposure Dosage
0%	-60	0.57	0.01	14.89	0.001	66.65	0.00	11.11
10%	-36	2.53	0.10	8.92	0.100	9.69	0.10	8.89
20%	-12	4.94	0.20	8.16	0.200	8.09	0.20	8.05
30%	12	24.28	0.30	7.83	0.300	7.56	0.30	7.65
40%	36	59.61	0.40	7.65	0.400	7.30	0.40	7.43
50%	60	136.78	0.50	7.53	0.500	7.15	0.50	7.29
60%	84	142.61	0.60	7.44	0.600	7.05	0.60	7.19
70%	108	142.61	0.70	7.38	0.700	6.97	0.70	7.12
80%	132	142.61	0.80	7.33	0.800	6.92	0.80	7.06
90%	156	142.61	0.90	7.29	0.900	6.88	0.90	7.02
100%	180	142.61	1.00	7.25	1.000	6.84	1.00	6.98

30-min Limit	Exposure Dosage	Slope	Exposure Dosage	Midpoint	Exposure Dosage	Scale	Exposure Dosage	Evacuation Implement
1	140.84	0.01	56.68	4	6.33	0.50	49.20	1
10	123.74	0.10	8.11	14	7.47	0.95	15.34	19
20	105.87	0.20	6.55	24	9.85	1.40	8.19	37
30	88.56	0.30	6.45	34	14.88	1.85	6.67	55
40	72.71	0.40	6.43	44	24.82	2.30	6.23	73
50	58.05	0.50	6.43	54	41.73	2.75	6.05	91
60	44.56	0.60	6.42	64	65.16	3.20	5.95	108
70	32.18	0.70	6.42	74	90.97	3.65	5.88	126
80	20.87	0.80	6.42	84	113.33	4.10	5.83	144
90	10.56	0.90	6.42	94	128.55	4.55	5.79	162
100	1.19	1.00	6.42	104	136.79	5.00	5.76	180

Exposure Dosage	Shelter Implement	Exposure Dosage	ACH	Exposure Dosage	Leakage Rate	Exposure Dosage	Respiratory Implement	Exposure Dosage
4.72	0	48.03	0.0	2.70	0.0	4.94	0	25.32
7.38	18	49.91	0.6	94.38	0.1	18.70	18	27.46
20.73	36	59.94	1.2	126.61	0.2	32.47	36	37.69
75.70	54	101.27	1.8	137.81	0.3	46.24	54	82.23
134.00	72	139.81	2.4	141.62	0.4	60.01	72	133.86
142.59	90	142.61	3.0	142.86	0.5	73.77	90	142.58
142.61	108	142.61	3.6	143.22	0.6	87.54	108	142.61
142.61	126	142.61	4.2	143.29	0.7	101.31	126	142.61
142.61	144	142.61	4.8	143.26	0.8	115.08	144	142.61
142.61	162	142.61	5.4	143.21	0.9	128.84	162	142.61
142.61	180	142.61	6.0	143.16	1.0	142.61	180	142.61

Figure E1. Sensitivity of exposure to warning system behavior parameter.

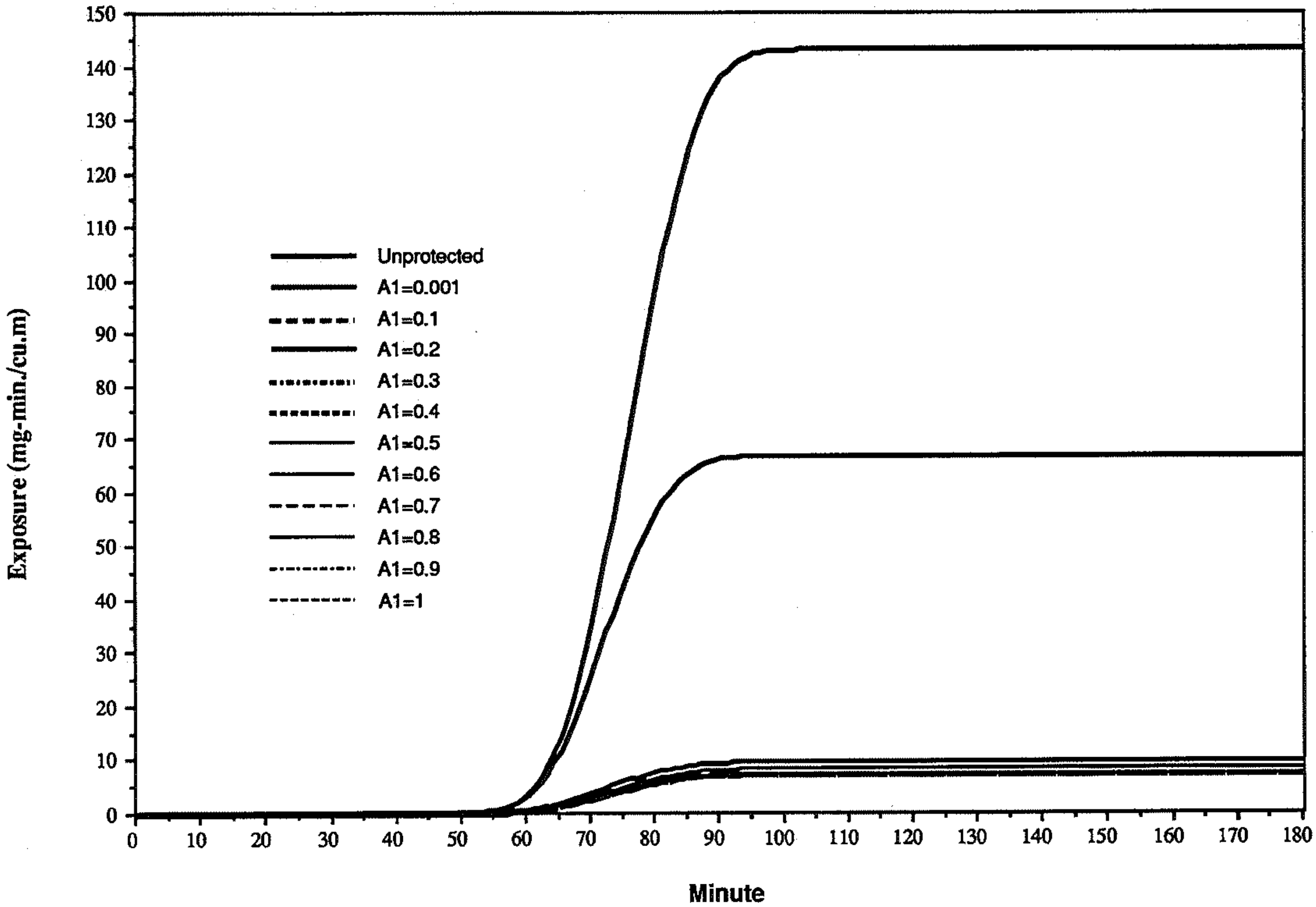


Figure E2. Sensitivity of exposure to warning system social

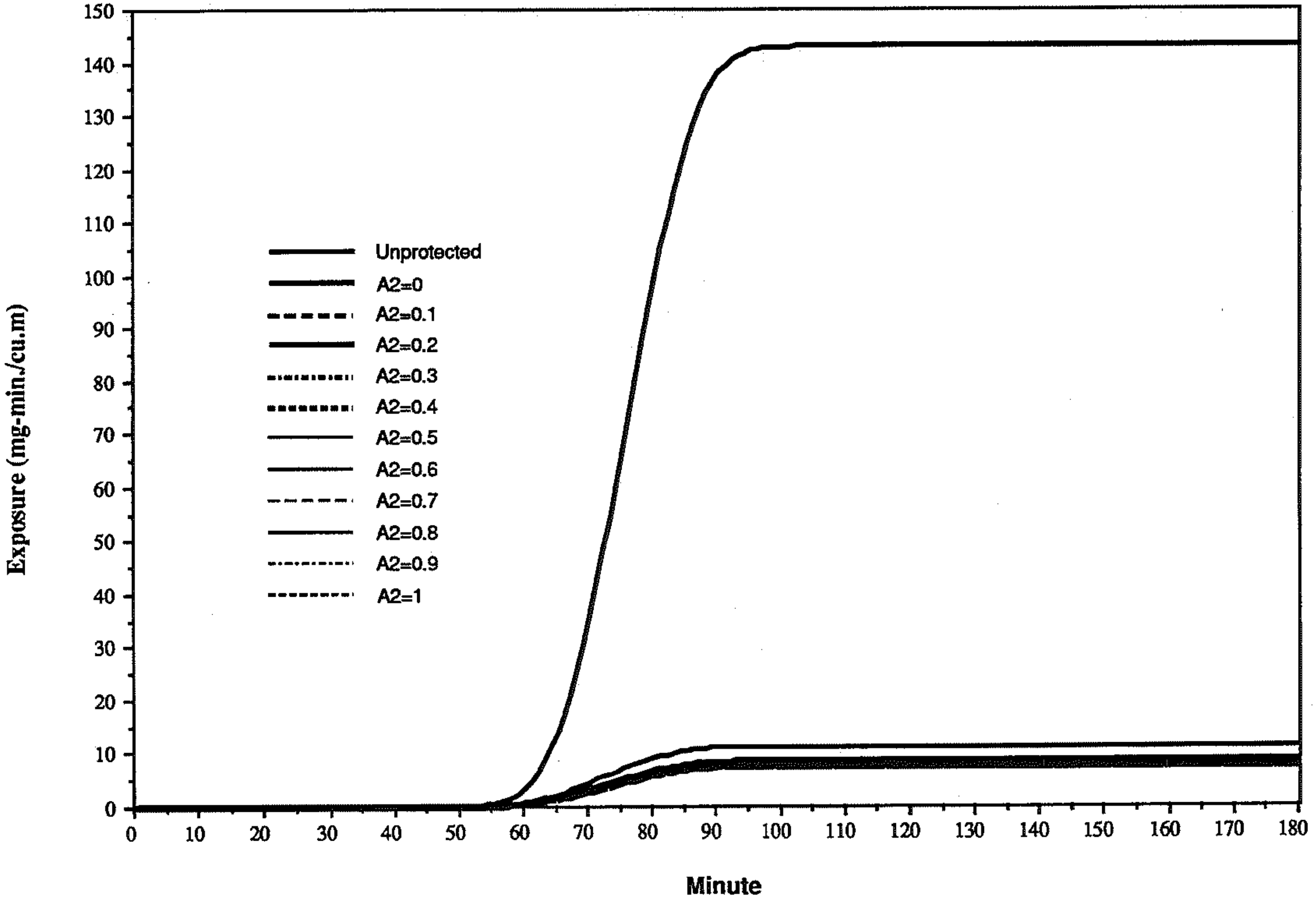


Figure E3. Sensitivity of exposure to warning system technological parameter

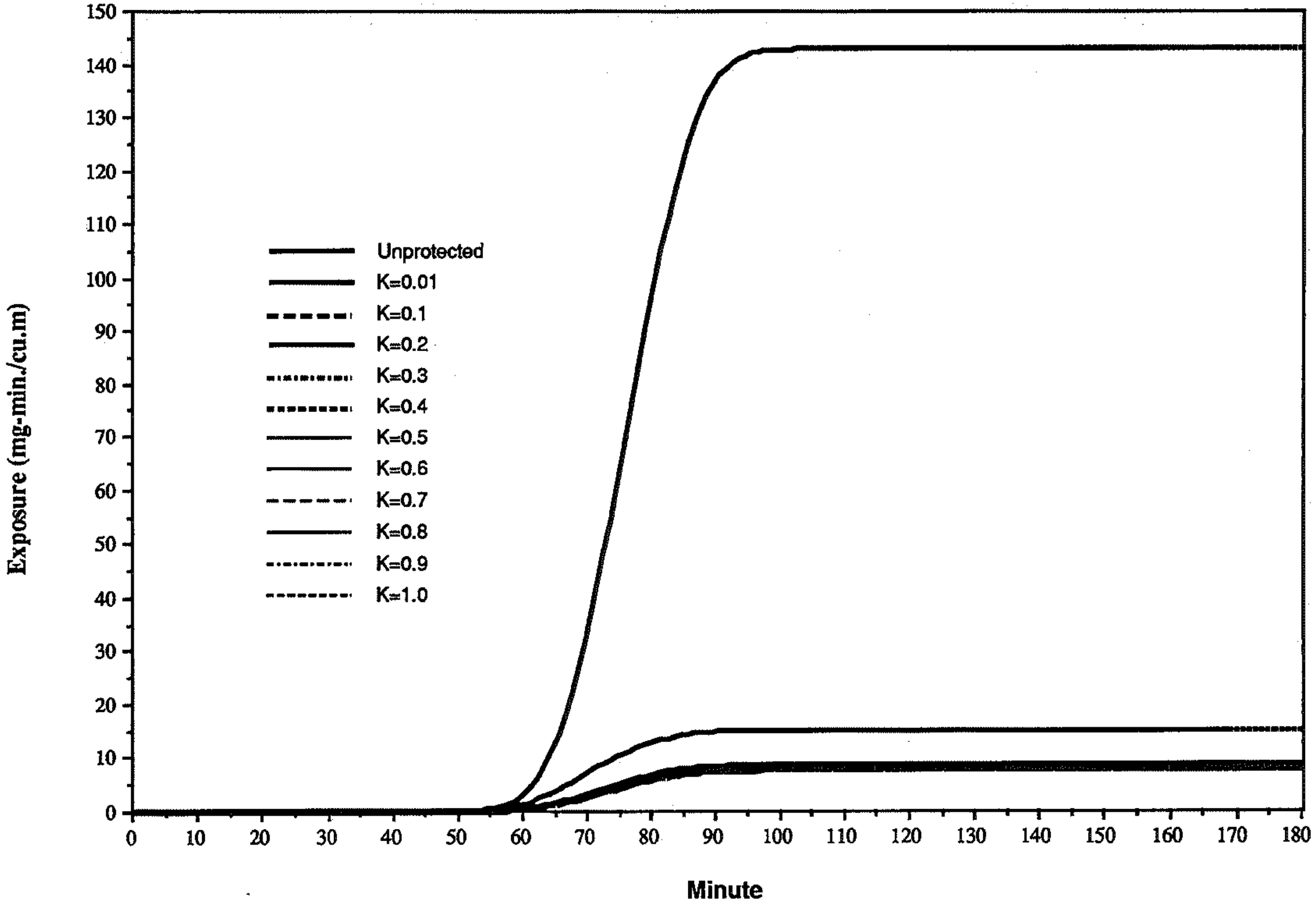


Figure E4. Sensitivity of exposure to warning system 30-minute limit.

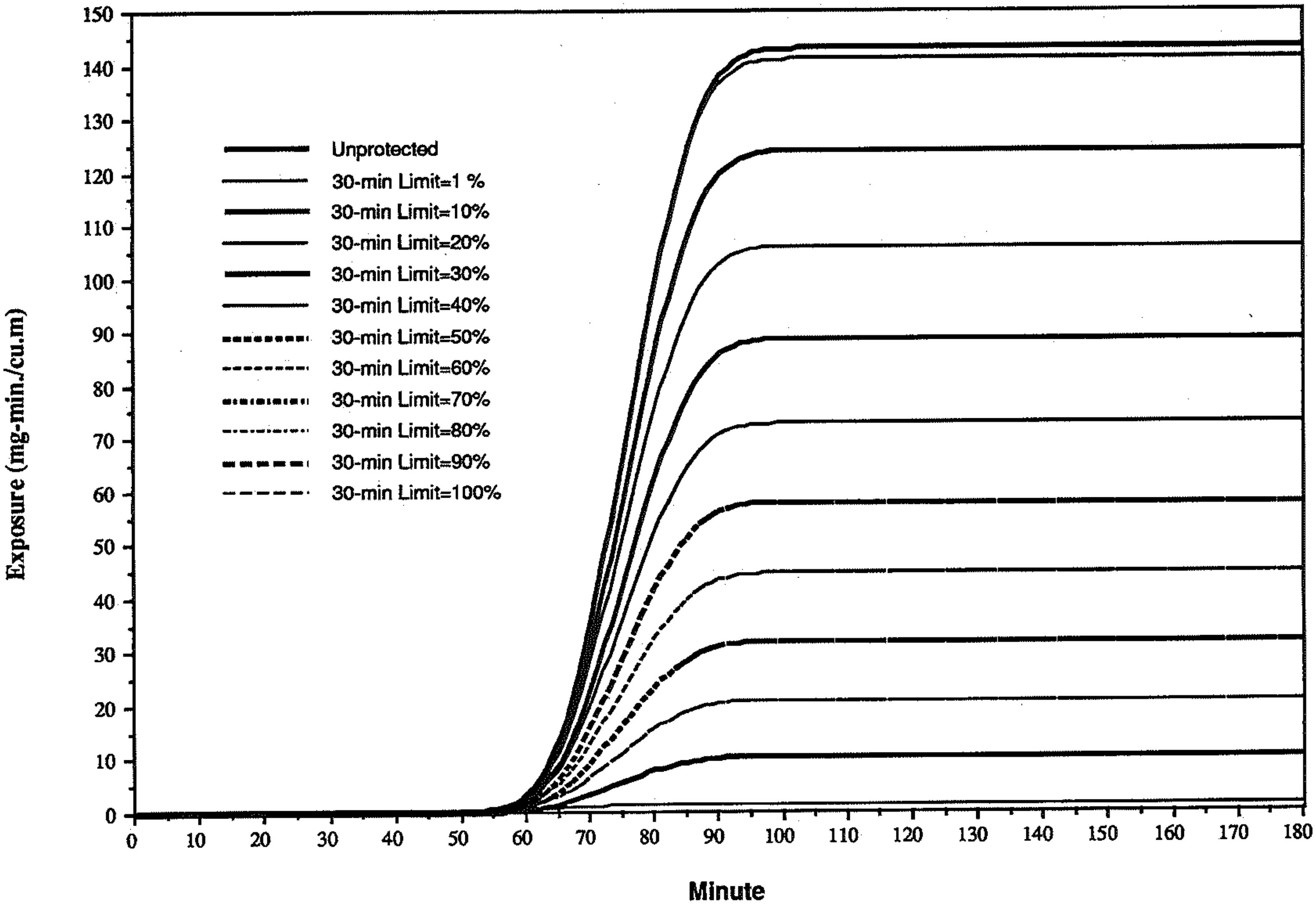


Figure E5. Sensitivity of exposure to warning system public response.

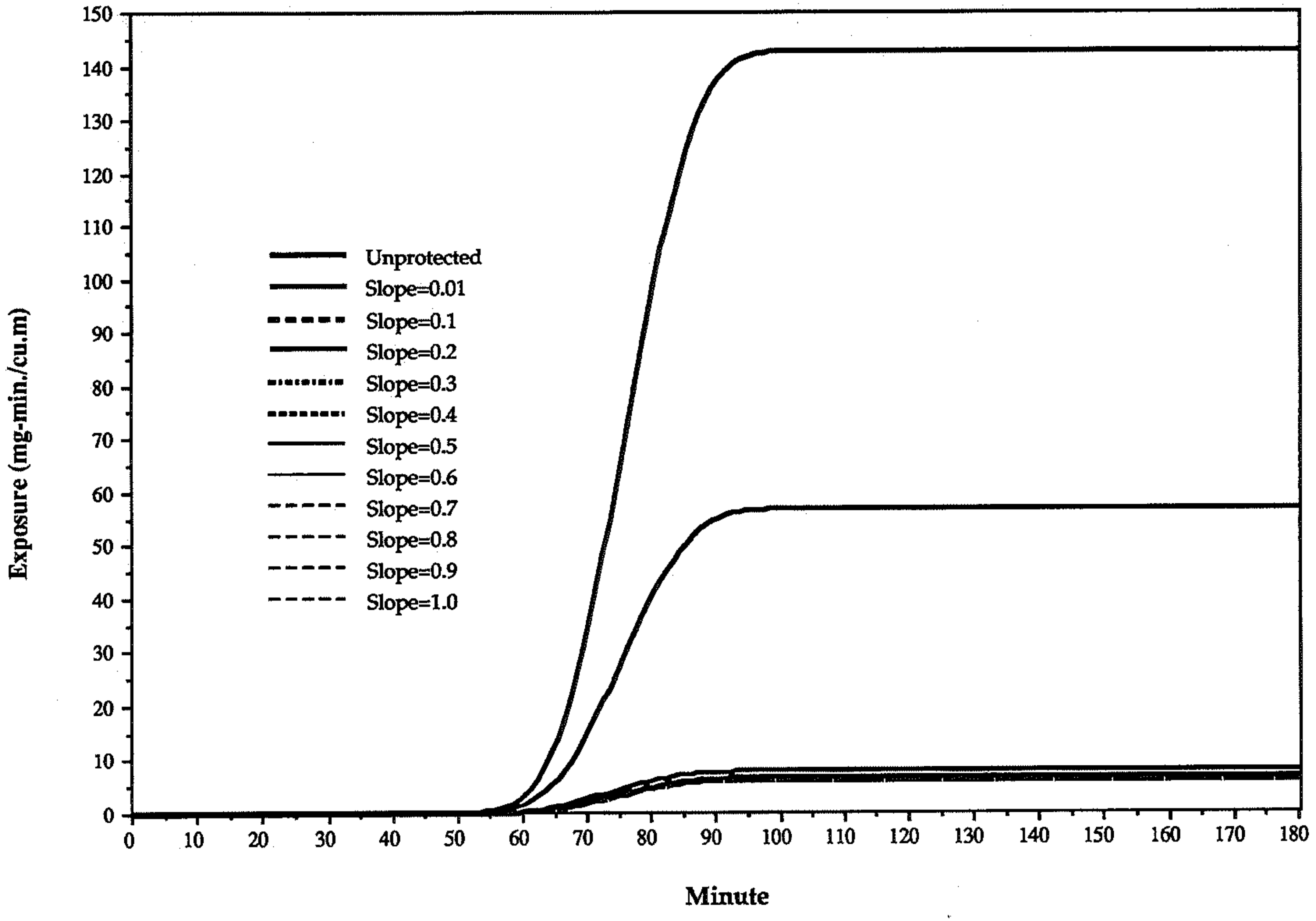


Figure E6. Sensitivity of exposure to warning system midpoint

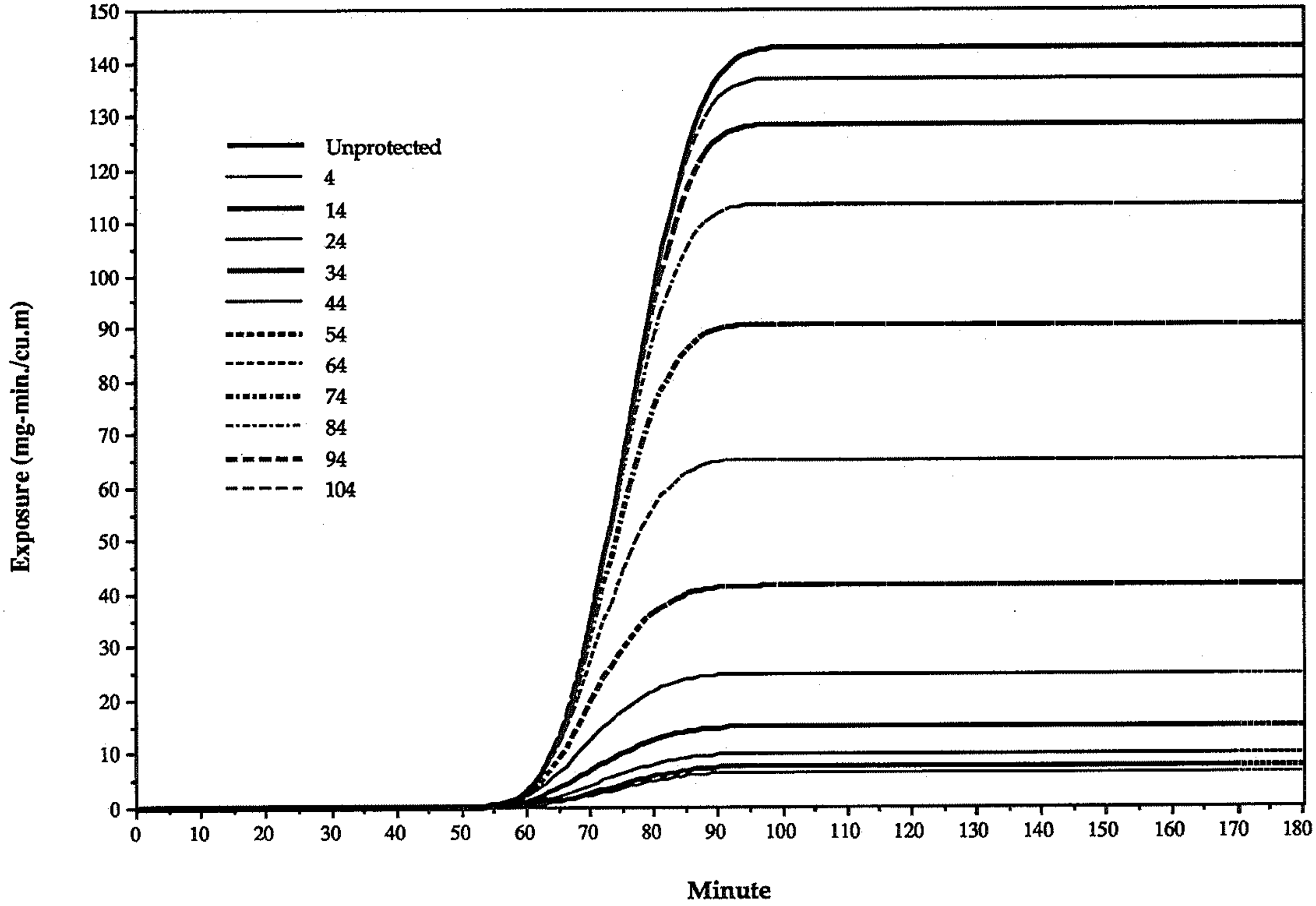


Figure E7. Sensivity of exposure to warning system scale

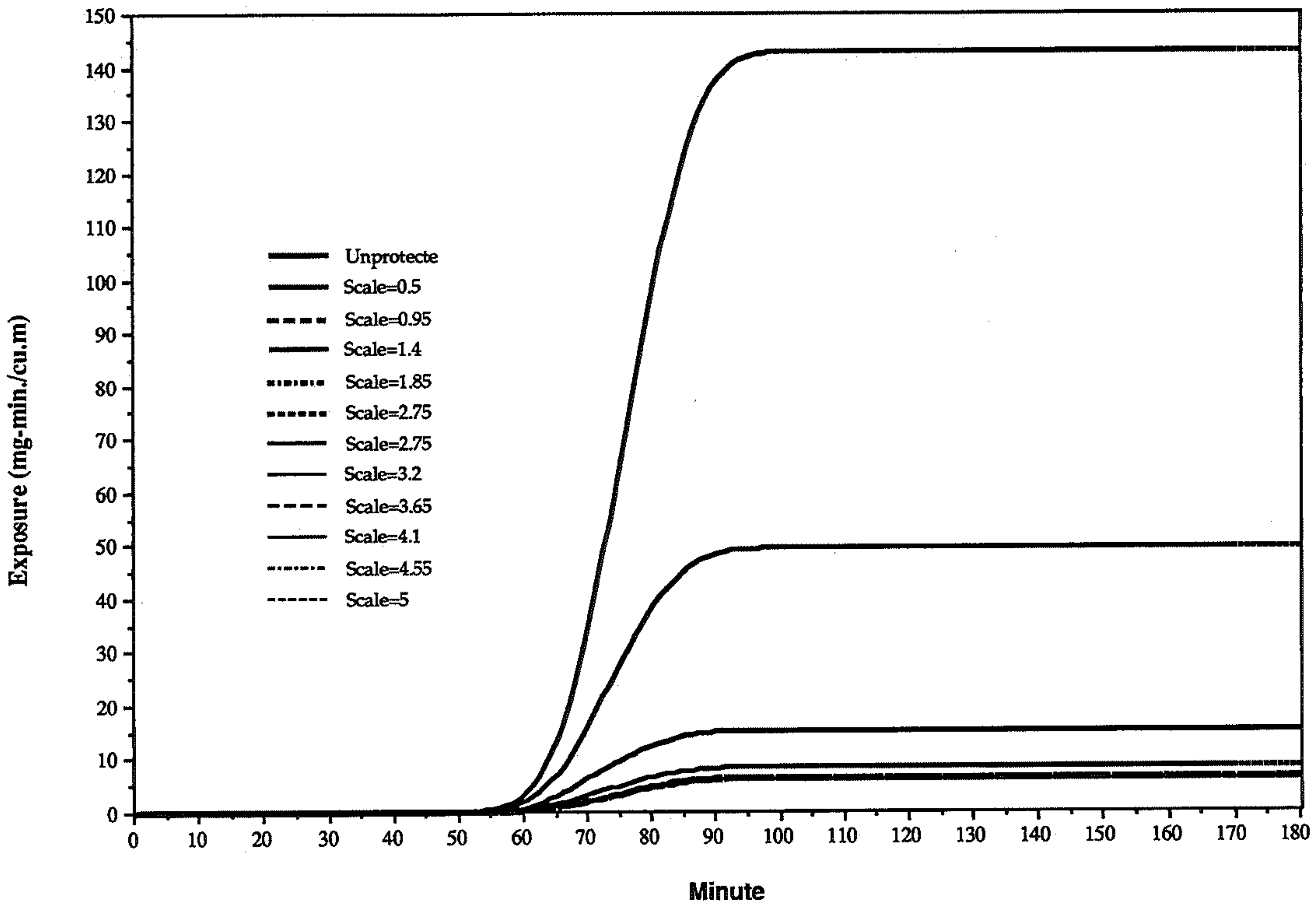


Figure E8. Sensitivity of exposure to warning system implementation time

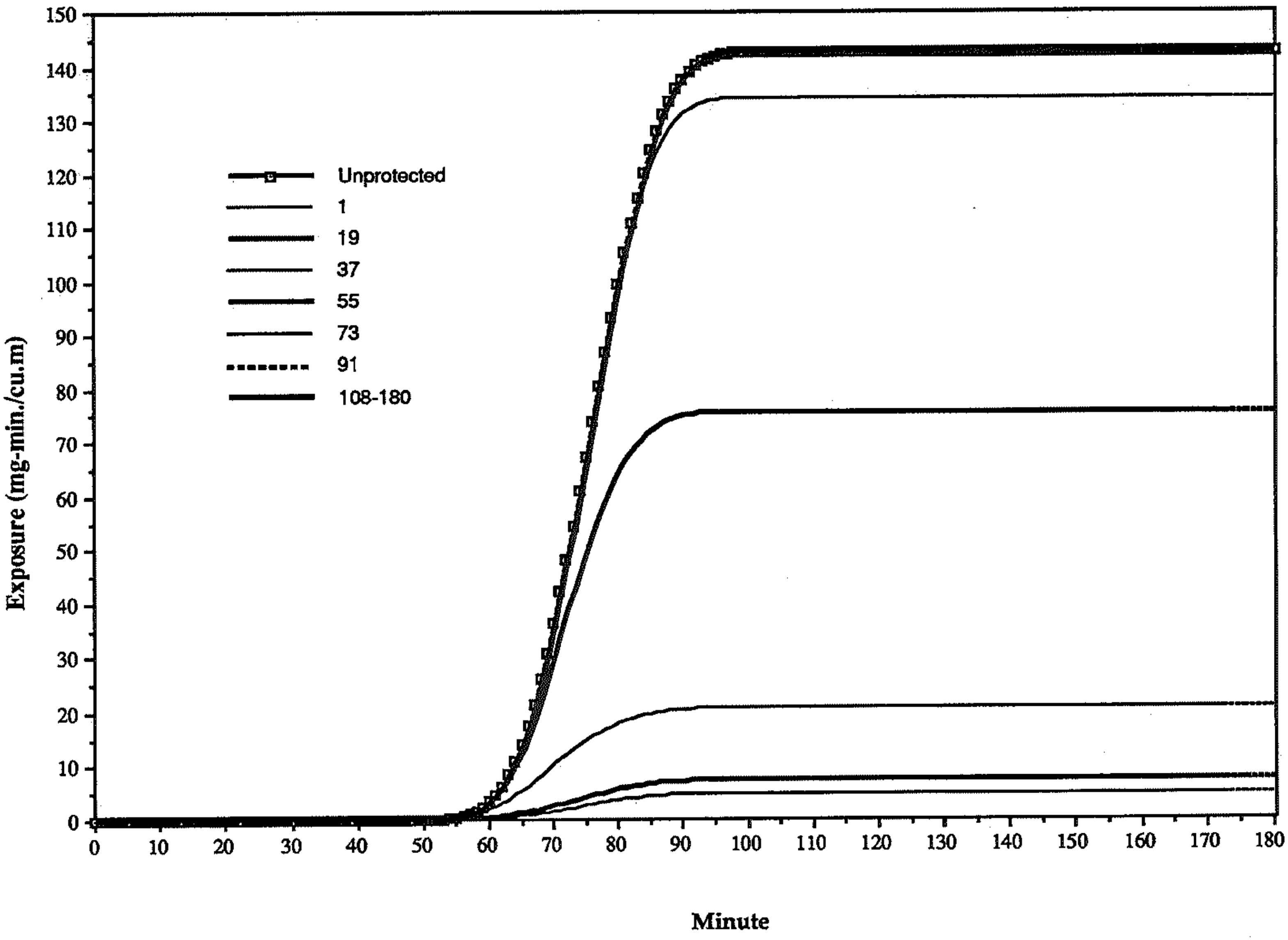


Figure E9. Sensitivity of exposure to warning system ACH

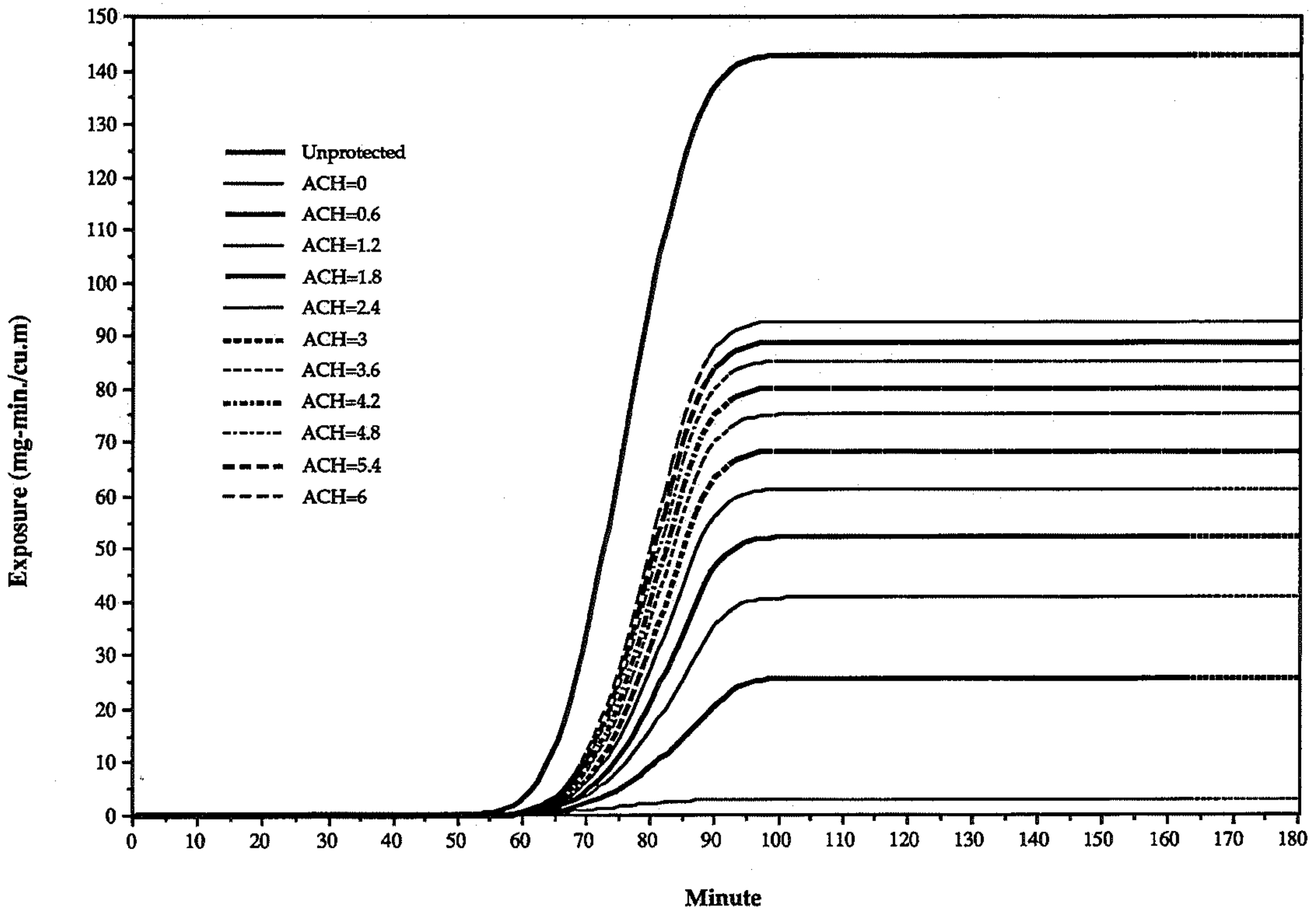
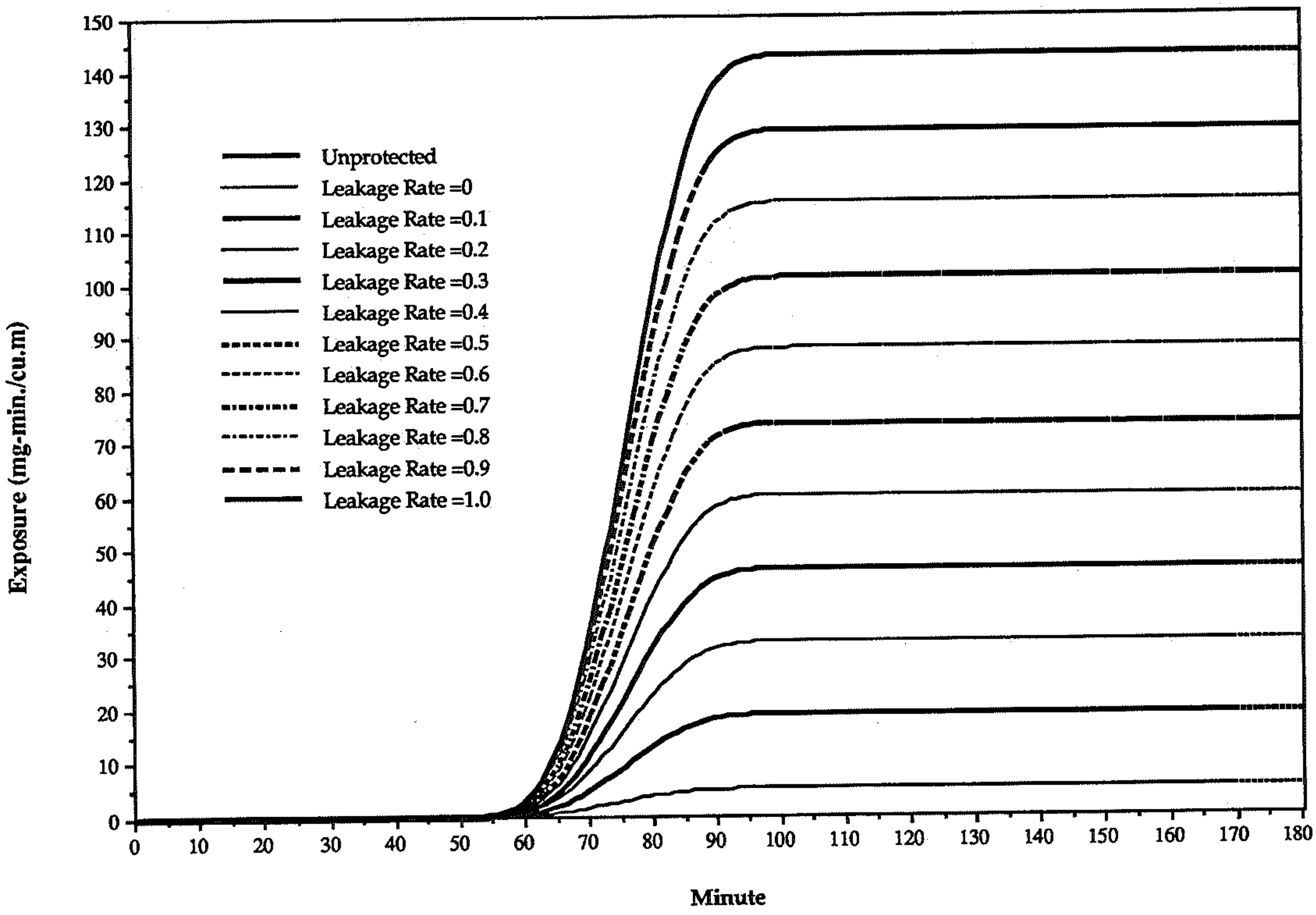


Figure E10. Sensitivity of exposure to warning system mask leakage rate



Appendix F: Sensitivity Analysis for Selected Parameters

By Jea-Guy Park and Jy-Pyng Sah

Sensitivity analyses for three different protection types are conducted with 13 parameters that change at 0-, 20-, 40-, 60-, 80-, and 100-percent scales while incident, meteorology and downwind distance are held constant. The incident involves 377 Kg of VX agent release at 6 AM and continues for 20 minutes when 1 m/s wind speed with E-stability and 500 m mixing layer height prevails. Warning is activated 5 minutes after release through tone-alert radios, sirens, and media/EBS warning systems. Public is located 4 Km from the incident site and expected to respond 50 percent better than Confluence curve and the implementation time in the public response window is set to zero. The 13 parameters vary from 0 to 100 percent with 20 percent increase. Each parameter's variation was determined by; 1) selecting mid-point between minimum value and maximum value in any predetermined systems for the parameter; 2) verifying that those minimum and maximum values were contained in the ± 50 percent of the mid-point range, and 3) verifying values for 0 to 100 percent with 20 percent increase. Table F-1 summarizes the parameter ranges tested for each parameter. In addition, we did full-range sensitivity analysis with the same scenario before mid-point sensitivity analysis. The full-range sensitivity analysis for each parameter was tested from minimum allowable value to maximum allowable value with 10 percent increase (cf. Appendix E).

Table F-1. Summary of Parameter ranges tested.

Parameter	0 (%)	20 (%)	40 (%)	60 (%)	80 (%)	100 (%)
Warning-Impl.	1	5	10	15	20	25
Technical	0.15	0.21	0.27	0.33	0.39	0.45
Behavior	0.137	0.192	0.247	0.302	0.357	0.412
Social	0.13	0.18	0.23	0.28	0.33	0.38
30min-Limit	0.3	0.44	0.58	0.72	0.86	1
Slope	0.03	0.05	0.07	0.09	0.11	0.13
Midpoint	10	15	20	25	30	35
Scale Factor	0.5	0.7	0.9	1.1	1.3	1.5
Evacuation- Clearance Time	10	26	42	58	74	90
Shelter-Impl.	1	12	24	36	48	60
Air change rate	0	0.45	0.9	1.35	1.8	2.25
Respiratory-Imp.	1	5	10	15	20	25
Leakage Rate	0	0.2	0.4	0.6	0.8	1

Note: Running Scenario:

Incident: Agent - VX 377 Kg; 55 percent Vaporized; Start of Release - 6 AM

Meteorology: Wind speed - 1.0 m/s; Mixing Layer Height - 500 m; Wind Stability Class - E.

Warning System: Activation After Release - 5 min; Method - TA Radios, Sirens, and Media/EBS.

Public Response: Distance - 4.0 Km; Response Curve - Confluence; Slope - 0.11; Midpoint - 15; Response Scale Factor - 1.50;

Protection: Evacuation Clearance Time - 20 min
In-Place Shelter Implementation - 0 min; Air Change Rate - 1.50
Respiratory Implementation - 3 min; Leakage Rate - 15 %

Figure F1. Limited range sensitivity of exposure to behavioral factor of warning system.

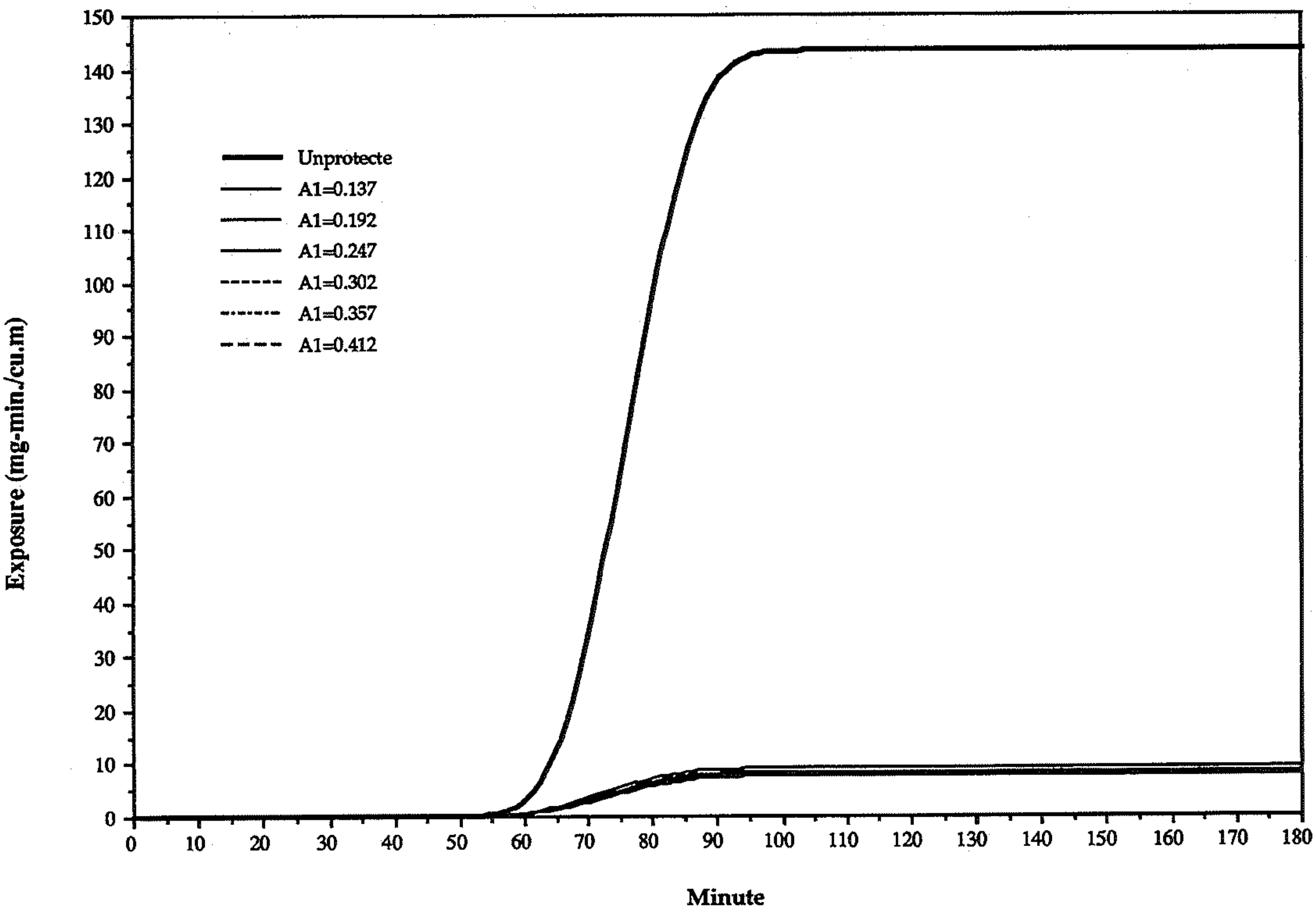


Figure F2. Limited range sensitivity of exposure to social parameter of warning system

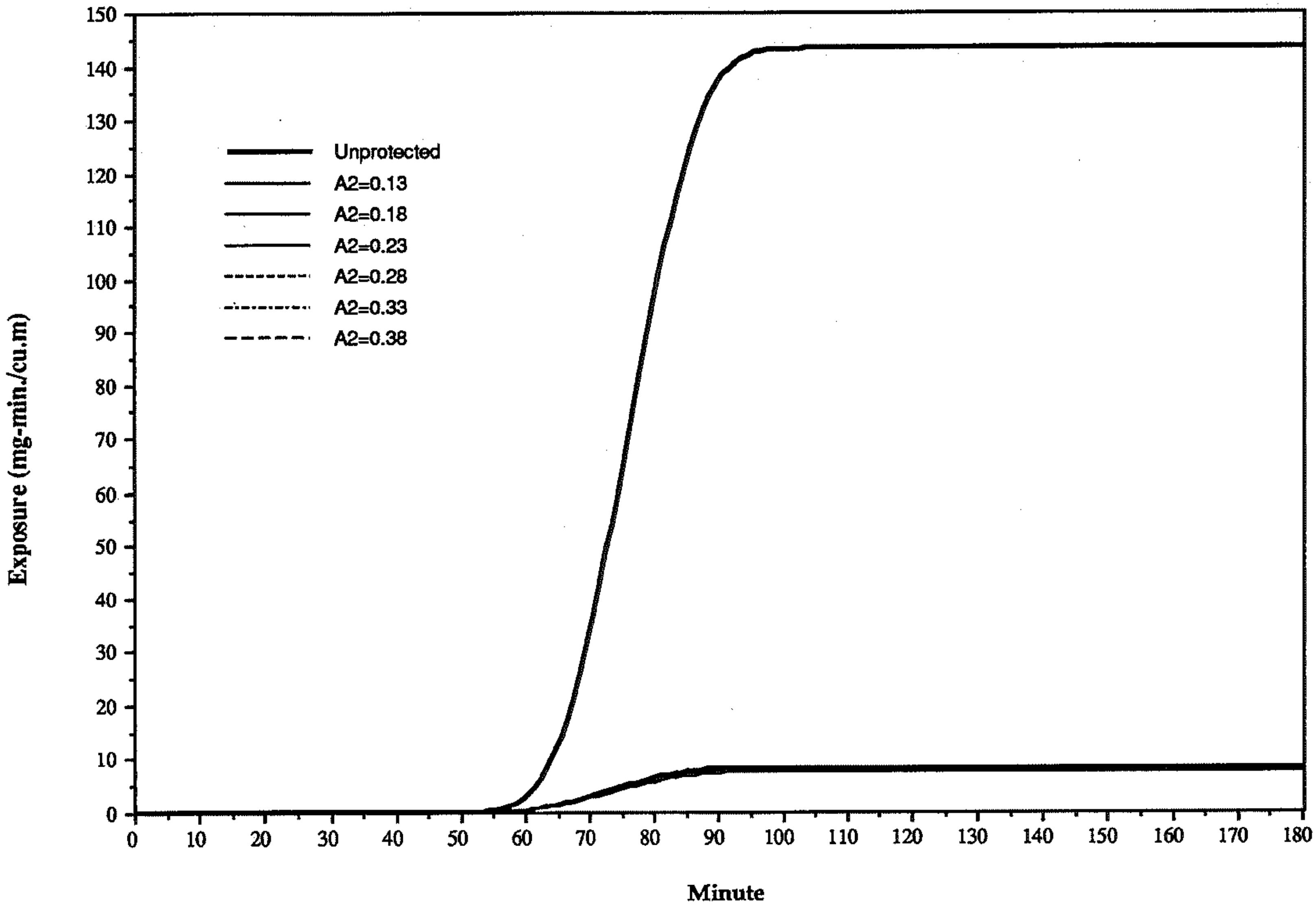


Figure F3. Limited range sensitivity of exposure to technological aspects of warning system.

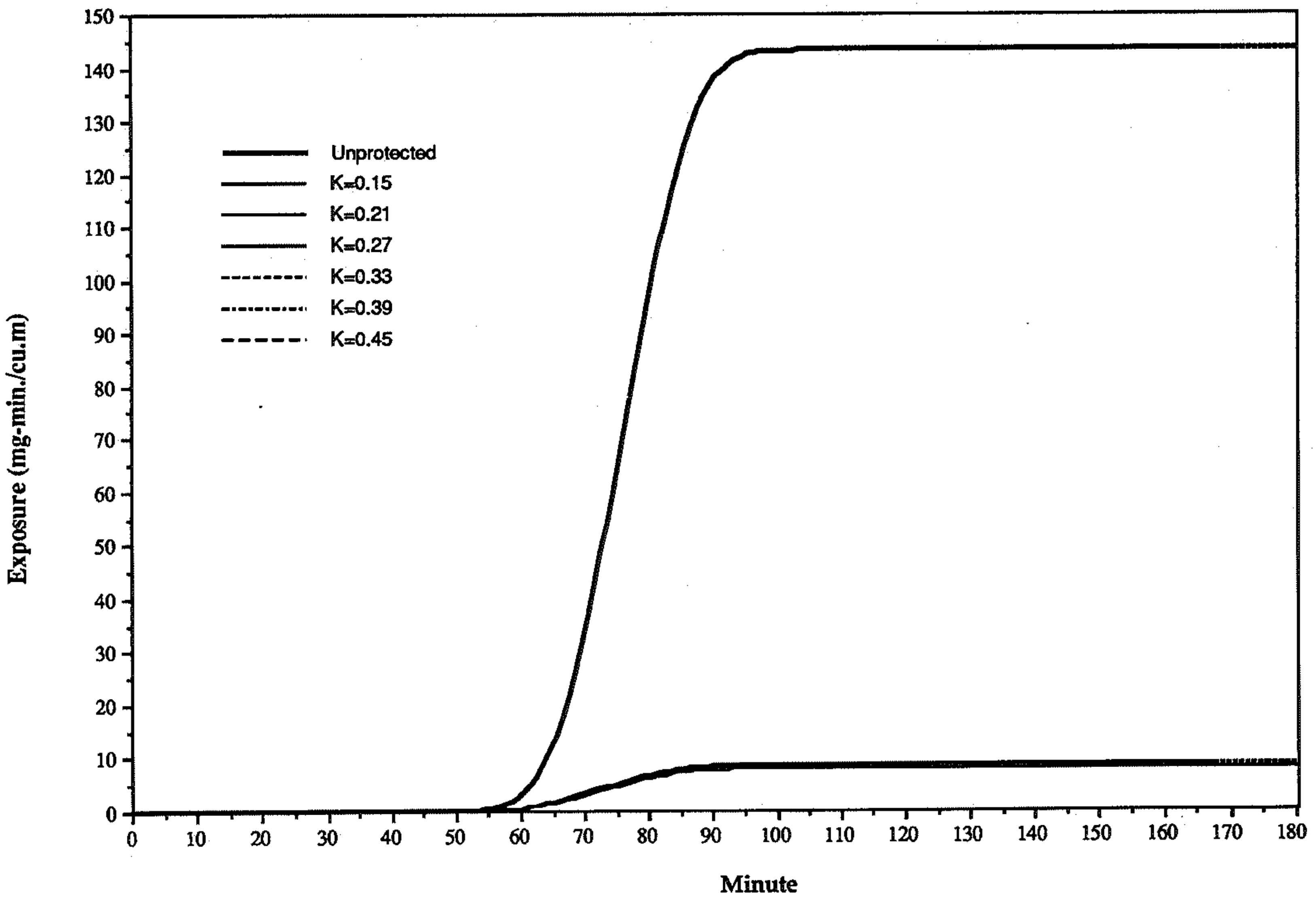


Figure F4. Limited range sensitivity of exposure to 30-minute limit.

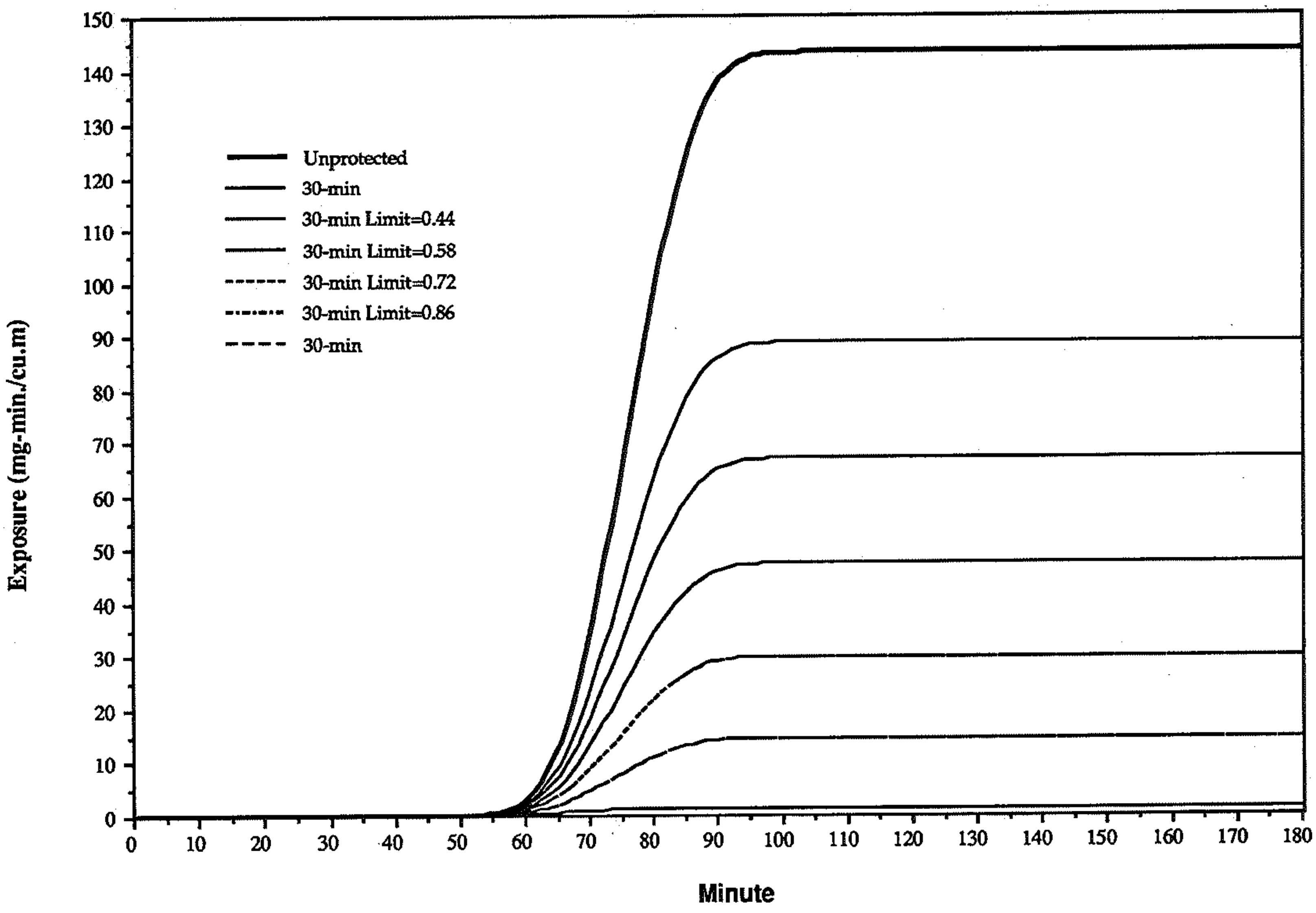


Figure F5. Limited range sensitivity of exposure to slope (public response).

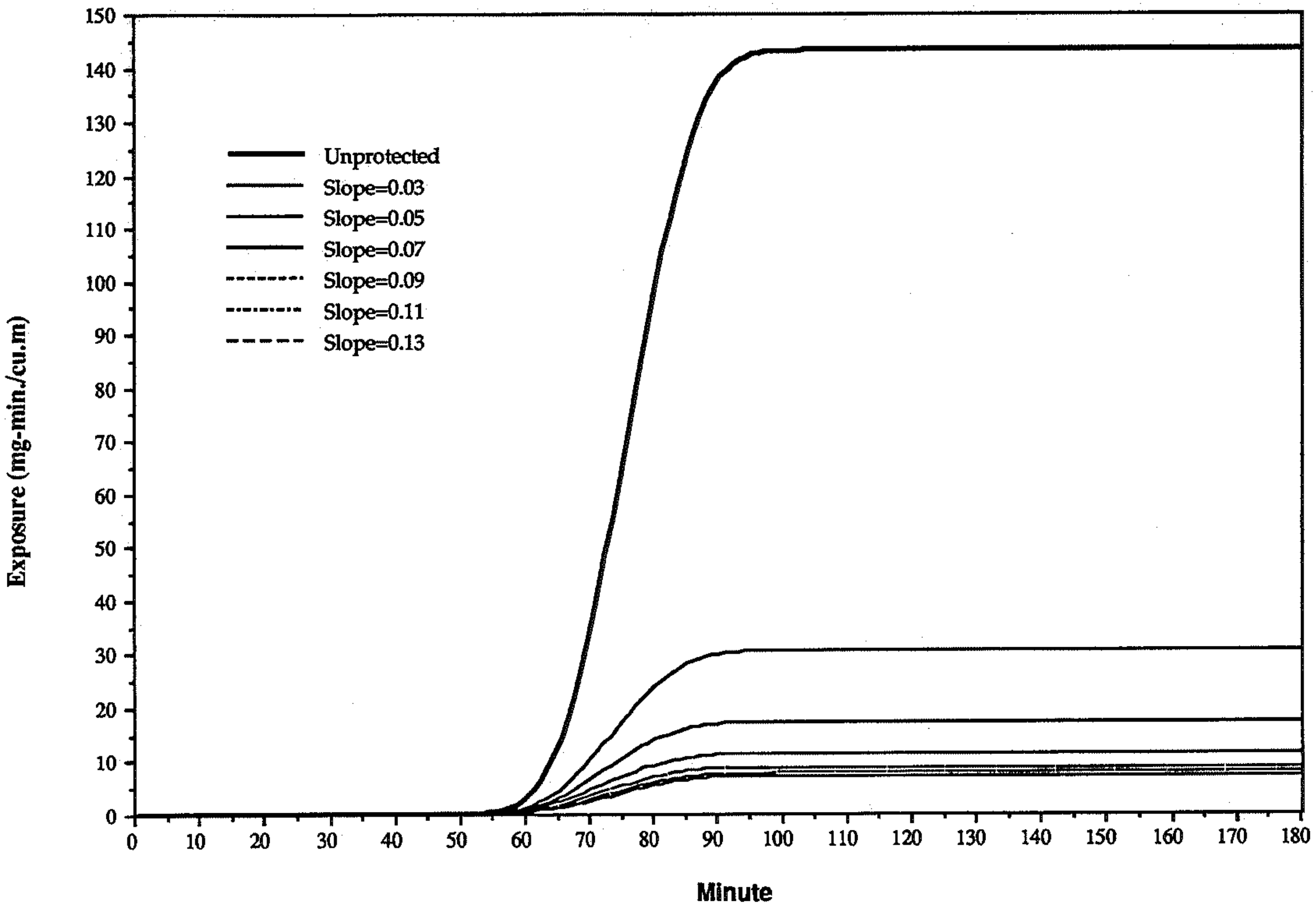


Figure F6. Limited range sensitivity of exposure curves to midpoint.

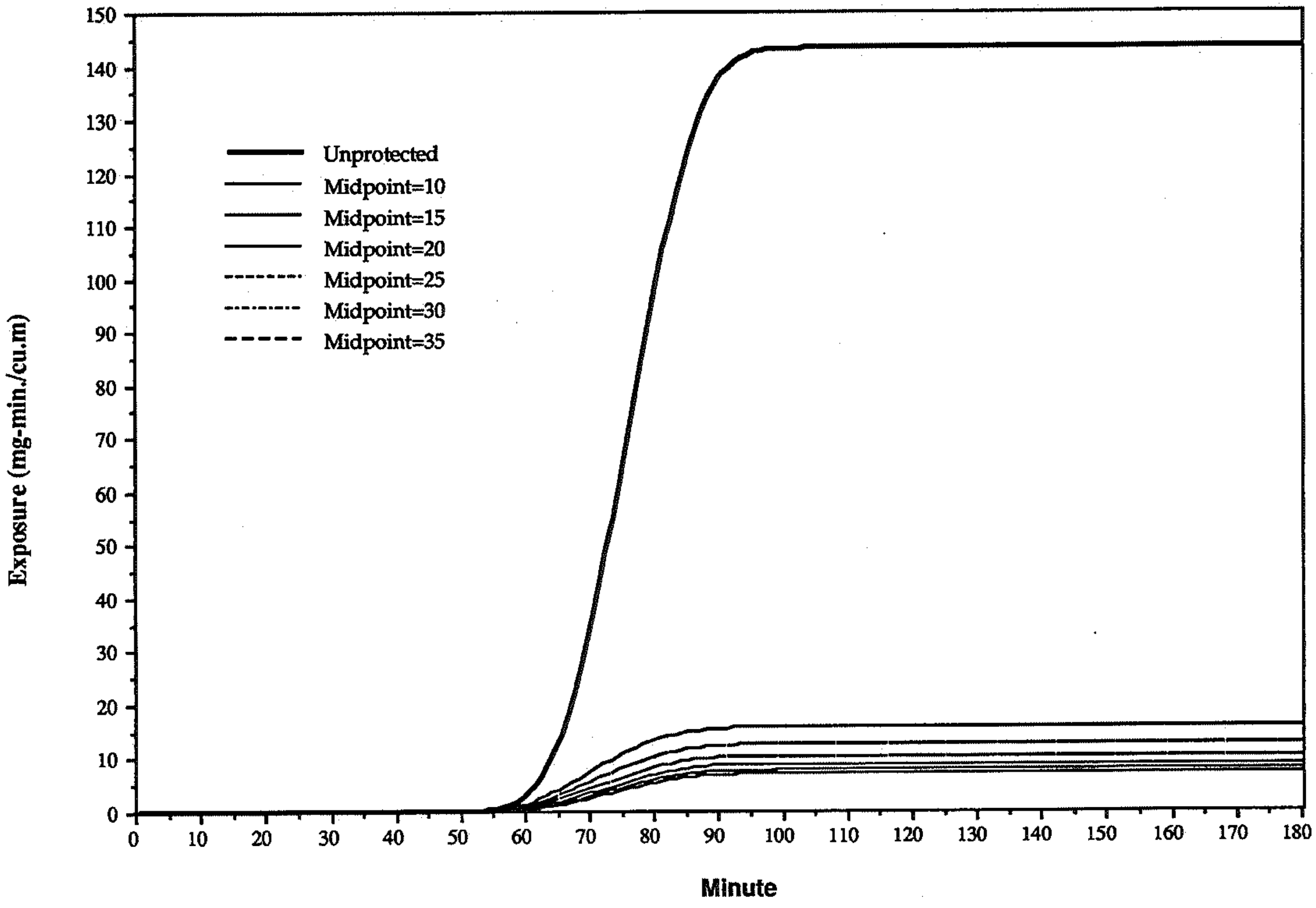


Figure F7. Limited range sensitivity of exposure curves to scale.

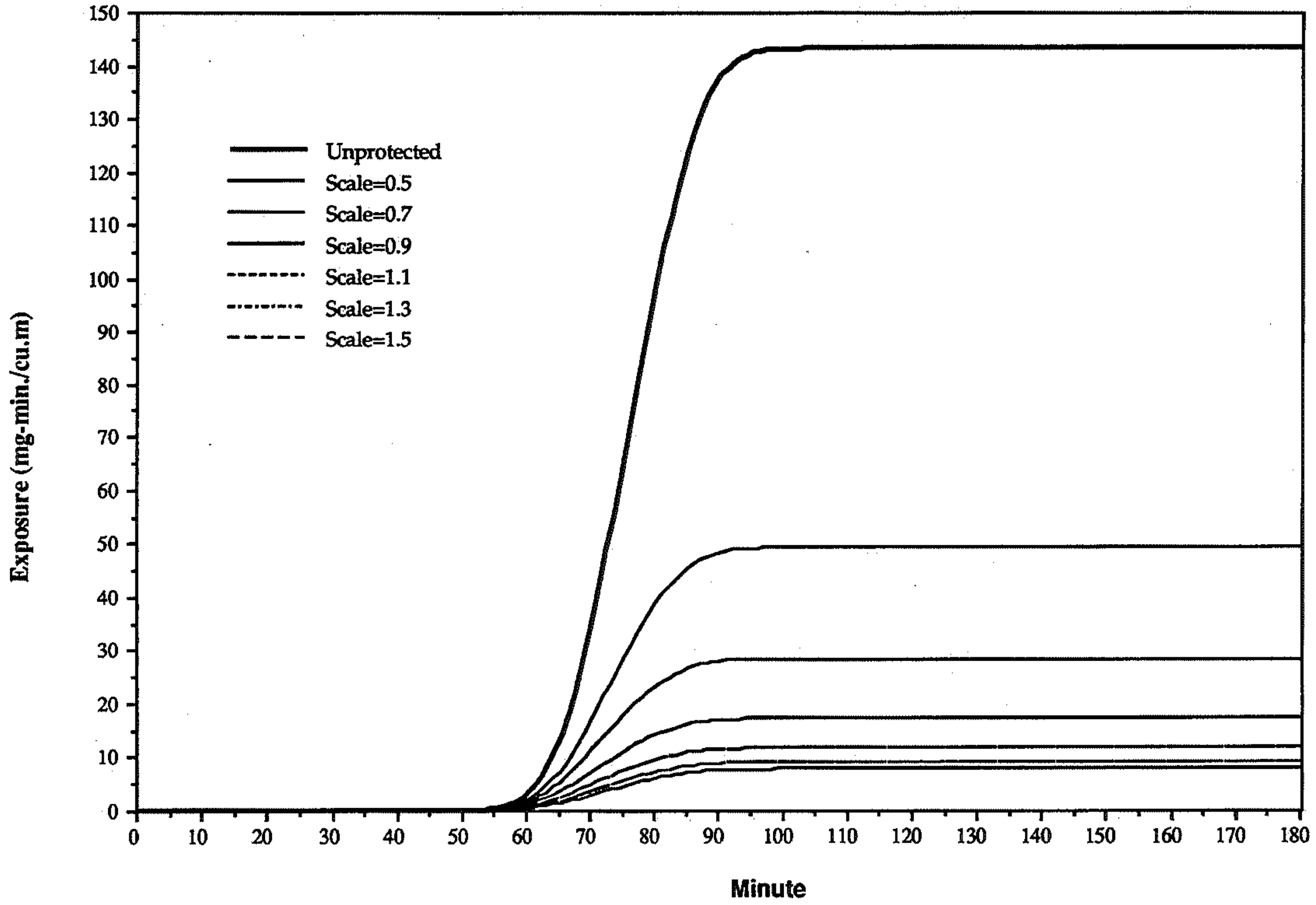


Figure F8. Limited range sensitivity of exposure curves to implementation time.

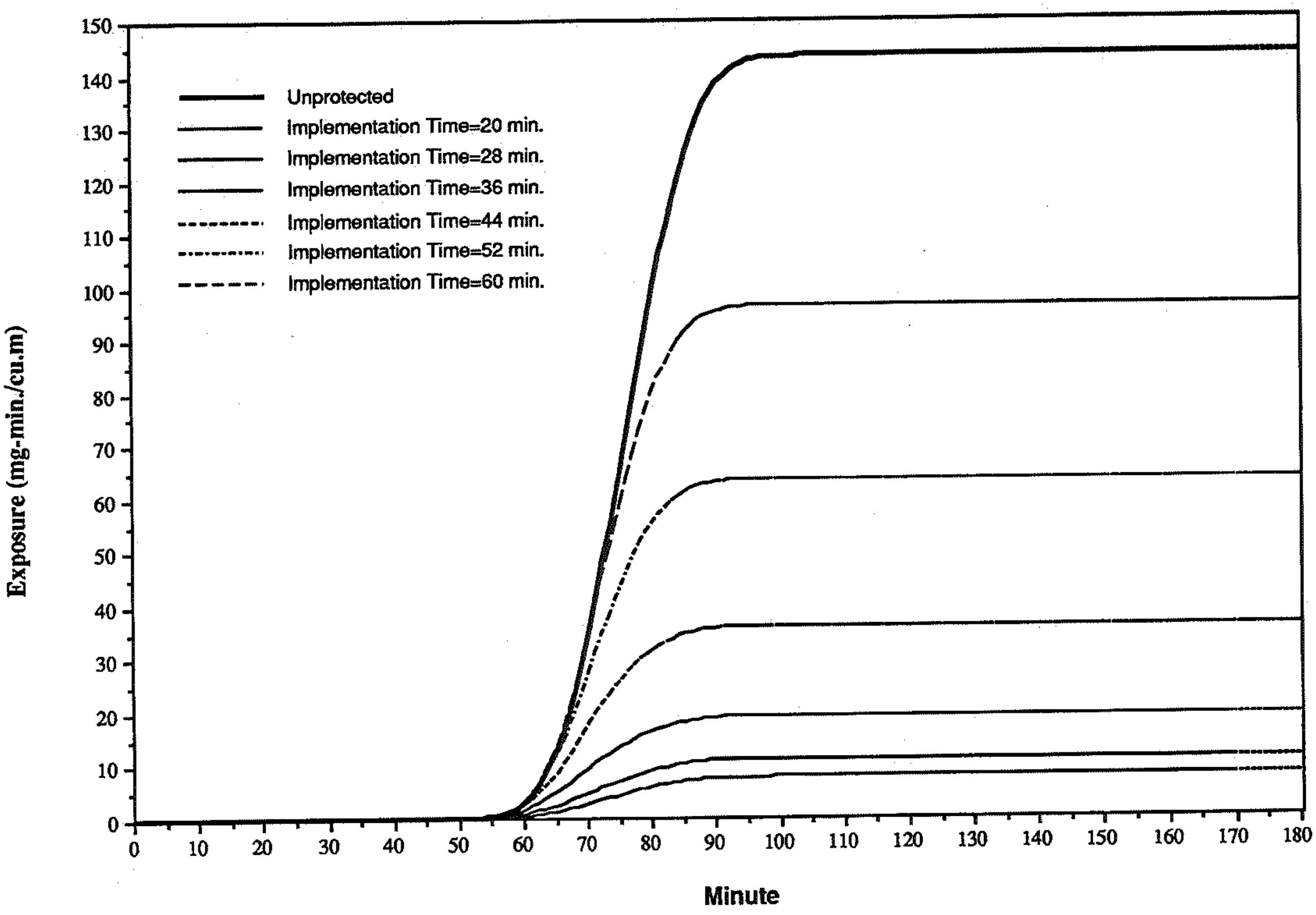


Figure F9. Limited range sensitivity of exposure curves to ACH.

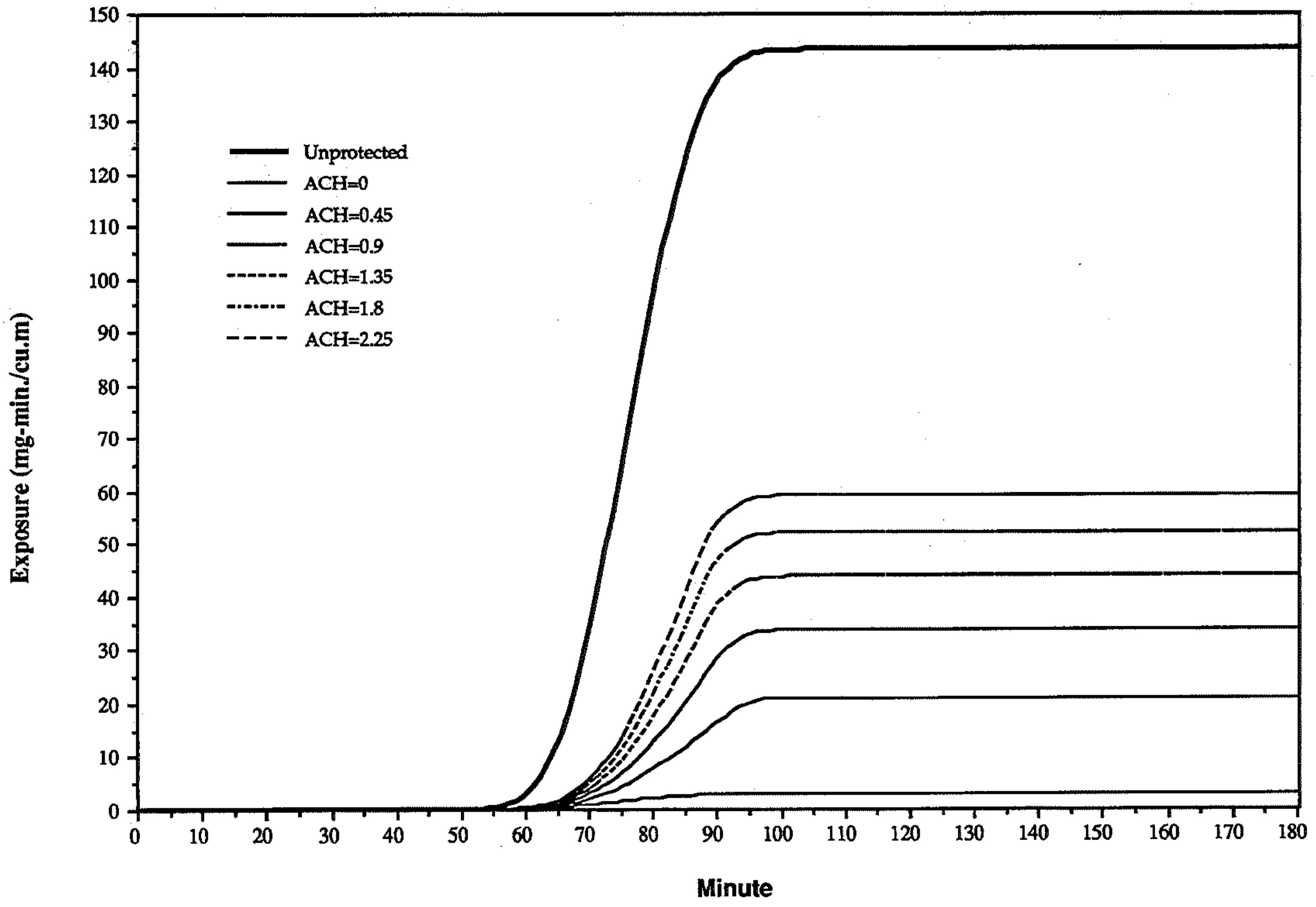
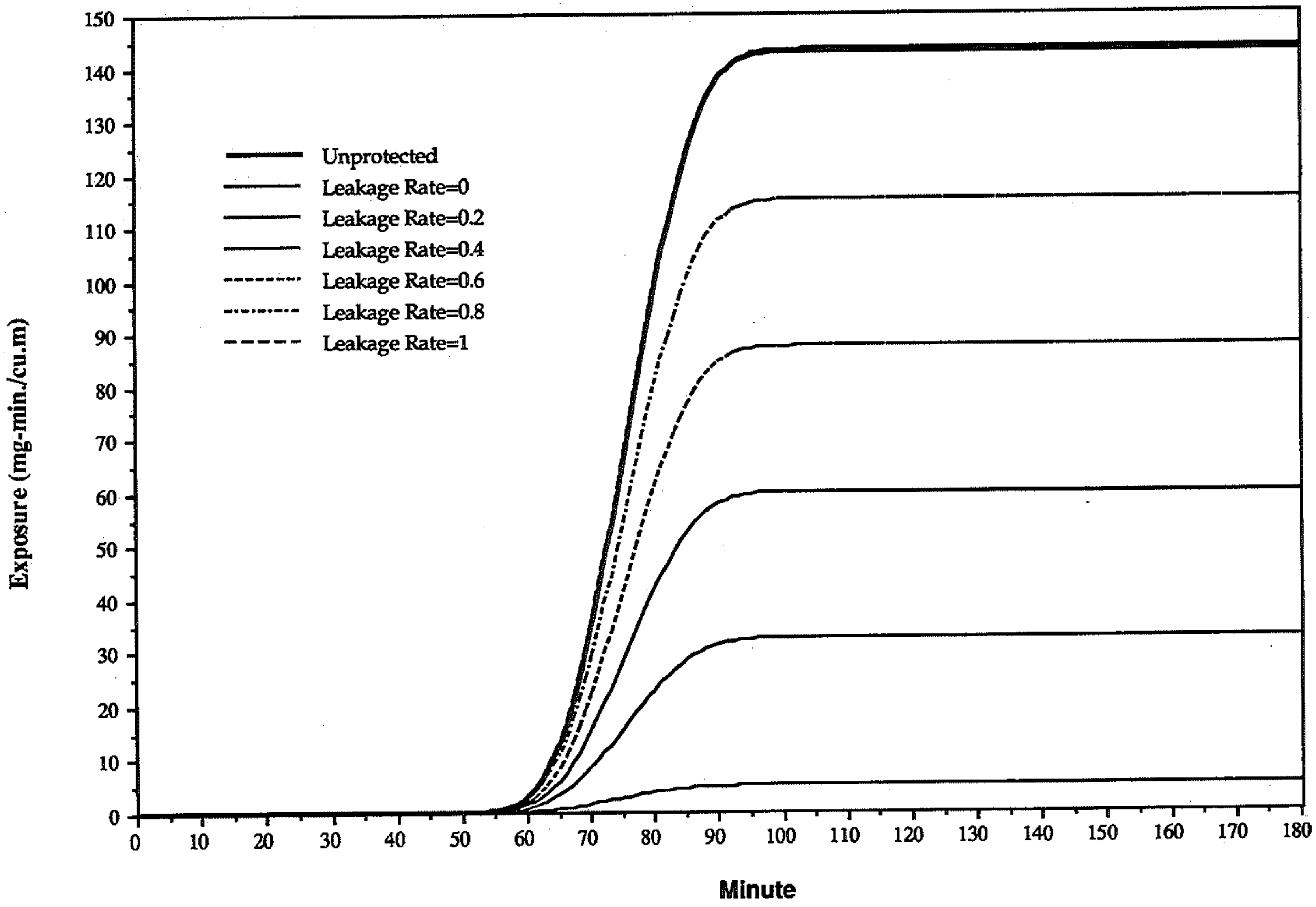


Figure F10. Limited range sensitivity of exposure curves to leakage rate.



Appendix G: Statistical Summary of Parameters in 600 Scenarios

By Jea-Guy Park and Jy-Pyng Sah

PADRE was run on 600 sample scenarios: 200 for each protection action. These runs were based upon a random normal distribution for all human response parameters. The range for the distribution varied for each factor (see Appendix F). Each of these scenarios involve various parameter changes in warning, public response, and protection implementation windows while incident, meteorology and downwind distance are held constant. The incident involves 377 Kg of VX agent release at 6 AM and continues for 20 minutes when 1 m/s wind speed with E-stability and 500 m mixing layer height prevails. Public is located 4 Km from the incident site. The scenario probability and exposure for each scenario were saved and randomly sampled. This sample consists of one case in every ten minutes of each scenario.

Table G-1. Statistical Summary of 200 Scenarios: Evacuation

	Minimum	Maximum	Mean	Variance	Standard Deviation
Activation (min)	0	60	29.2	284.4	16.9
Technological	0.13	0.38	0.249	0.006	0.077
Behavioral	0.150	0.400	0.2797	0.0048	0.0694
Social	0.15	0.45	0.297	0.007	0.086
30-Min Limit	31	99	64.8	392.5	19.8
Slope	0.04	0.12	0.079	0.001	0.024
Mid-Point	10	35	23	55.8	7.5
Scale	0.50	2.49	1.426	0.361	0.601
Implementation (min)	10	90	49.7	532	23.1

Table G-2. Statistical Summary of 200 Scenarios: In-Place Shelter

	Minimum	Maximum	Mean	Variance	Standard Deviation
Activation (min)	1	60	28.8	293.3	17.1
Technological	0.13	0.38	0.256	0.006	0.075
Behavioral	0.151	0.400	0.2737	0.0054	0.0738
Social	0.15	0.45	0.305	0.007	0.086
30-Min Limt	30	100	65.4	412.6	20.3
Slope	0.04	0.12	0.081	0.001	0.024
Mid-Point	10	35	22.3	51.3	7.2
Scale	0.51	2.50	1.507	0.312	0.558
Implementation (min)	1	45	22.4	153	12.4
ACH	0.01	2.48	1.208	0.572	0.757

Table G-3. Statistical Summary of 200 Scenarios: Respiratory

	Minimum	Maximum	Mean	Variance	Standard Deviation
Activation (min)	0	60	29.7	306	17.5
Technological	0.13	0.38	0.248	0.005	0.073
Behavioral	0.150	0.400	0.281	0.005	0.073
Social	0.15	0.45	0.294	0.007	0.086
30-Min Limt	31	99	64.2	393	19.8
Slope	0.04	0.12	0.081	0.001	0.023
Mid-Point	10	35	22.6	48	6.9
Scale	0.51	2.50	1.469	0.318	0.564
Implementation (min)	1	10	5.9	6.3	2.5
Leakage (%)	5	40	23.5	105.8	10.3

Note: Running Scenario:

Incident: Agent - VX 377 Kg; 55 percent Vaporized; Start of Release - 6 AM

Meteorology: Wind speed - 1.0 m/s; Mixing Layer Height - 500 m; Wind Stability Class - E.

Warning System: Activation After Release - 5 min; Method - TA Radios, Sirens, and Media/EBS.

Public Response: Distance - 4.0 Km; Response Curve - Confluence; Slope - 0.11; Midpoint - 15; Response Scale Factor - 1.50;

Protection: Evacuation Clearance Time - 20 min
In-Place Shelter Implementation - 0 min; Air Change Rate - 1.50
Respiratory Implementation - 3 min; Leakage Rate - 15 %

Appendix H: W. Naegeli (ORNL) comments and Author Response about Emergency Response

By ORNL and George Rogers

W. Naegeli (ORNL) comments: The principal purpose of PADRE is to serve as a planning tool for determining what would be the optimal protective-action recommendation for a given set of conditions. PADRE helps the emergency planner to systematically analyze and categorize incident scenarios such that an appropriate protective-action recommendation can be chosen quickly in the event of an actual release of chemical agent.

In case of an emergency, the matching of actual release and meteorological conditions with pre-analyzed scenarios will make it possible for the decision maker to reduce the dosages received by the public by focusing on two principal decision components: when to warn the public and what protective action to recommend. The appropriateness of the recommendation depends to a large extent on the amount of time available between the activation of the warning system and the arrival of the chemical-agent plume. These factors determine what proportion of the public receives the warning in time to complete implementation of the protection. Thus the success of the recommendation depends directly on what moment in time the warning system gets activated.

Of course the success of the recommendation depends indirectly on how long it takes to reach a decision, but only because that determines the time of warning-system activation, which is the endpoint of the decision phase. In practice, the decision maker must select the protective-action recommendation in the knowledge of the time when the warning system will be activated. If the decision process takes too long, it may preclude the choice of evacuation as the recommended action. *Dosage reduction is a function of the warning-system activation time. It needs no knowledge of the decision phase.*

PADRE does *not* use the warning-system activation parameter to suggest that emergency planners "mandate the amount of time required by emergency responders and decision-makers." This parameter lets planners perform what-if analyses that assess how the timing of the warning affects the dosage reductions achievable with different protective actions. Indeed, such analyses may be used to determine whether installation of a more effective a warning system could give the decision makers extra time in most-likely or worst-case scenarios.

Of course, by varying the activation-time parameter, the planner also can demonstrate to the decision makers how the length of time needed to reach a decision is likely to affect the outcome of the emergency. If anything mandates the timing of the decision process, it is the combination circumstances of the chemical-agent release, not the planner. The planner should use PADRE to tell the decision makers about windows of opportunity that might rapidly be missed.

That knowledge is crucial for them to reach an informed decision and may make it more likely for the decision to be reached in a timely manner.

It certainly could be useful for PADRE to assist emergency planners in analyzing and improving the decision-making process itself. However, under the current architecture of PADRE, such a capability is not a necessity for computing good estimates of dosage reductions.

PAECE, a precursor to PADRE, includes a probability function for the decision to warn. PAECE is a research tool. It provides options to stochastically analyze classes of emergency scenarios. It helps scientists to better understand and characterize the nature and probable range of chemical emergencies.

By contrast, PADRE is a planning tool. It is designed for the systematic analysis of series of individual scenarios to help the planner identify threshold conditions that require different protective-action recommendations. In the event of an emergency, it is important to base the protective-action recommendation on the exact timing for the actual incident, rather than the theoretical probability of when a decision might be reached. PADRE was designed primarily as a planning tool and secondarily as an aid for validating the response and assessing the impact of changing conditions during actual emergencies. To simplify the user interface and reduce the risk of operator errors, we did not implement in PADRE the research features of PAECE.

A future version of PADRE may incorporate additional features, such as tools for analyzing the decision process. These tools possibly could take the form of separately loadable modules that will not distract the operator during routine or emergency use of the model. *To be effective in practical emergency-planning, a tool to analyze the decision process should be based on the evaluation of empirical evidence as well as a comprehensive theoretical model of emergency decision making* . Conceptually, such a tool might allow the user to configure input parameters for the number of people involved in the decision, their roles and relationships to each other, their experience and understanding of chemical emergencies, the degree of urgency to reach a decision, the magnitude of the threat, the likelihood of each player to be reachable during an emergency, the probable timing of his or her entry into the decision-making process, and possibly one or more psychological parameters for personality traits that might be predictive of the capacity to reach a decision under stress. While the latter parameters may be significant, they would be difficult to assess in a civilian environment with elected officials responsible for the decision.

Author Response: There is both considerable agreement and some legitimate disagreement on this point. We certainly agree that using the minute warning system activated allows the emergency planner to examine individual scenarios effectively, conveniently and without error. In short, there is no theoretical conflict. Moreover, to the extent that community emergency planners do not control the decision-making process letting the process begin with a simple minute-begin parameter is both logical and theoretically consistent. The legitimate disagreement is both subtle and indirect. The theoretical models underlying PADRE are probabilistic in that they examine a distribution of empirical cases in terms of the underlying theory. Because PADRE is comprehensive, some of the models are based on larger distributions than others. In addition, some of the processes (e.g., infiltration in a leaky shelter) are more deterministic in nature than others (e.g., warning). PADRE currently summarizes both the deterministic and probabilistic information in terms of central tendency. This is perfectly appropriate for both deterministic and probabilistic models, but for probabilistic models it tends to under-emphasize the variability of the underlying distribution. Ideally, when the knowledge base for PADRE is probabilistic, the input-output should reflect that distributional quality in terms of both central tendency and variance. PARDE's output was very carefully designed to reflect a distributional quality, by bounding the expected value results with capacity to protect and unprotected exposure expectations, and future versions of PADRE may want to move further in this direction.

The empirical base for decision to warn theory is somewhat limited in that only 70 accident/incidents have been studied to date. These cases, while better than no empirical evidence, are mostly limited to private industries' relationship with small communities. And many of these accidents involved transportation and thereby did not exhibit extensive advance relationships with the community involved. The CSEPP site are quite different in this regard. The military-community relationship is both on-going and extensive. In addition, the amount of specific training for both military and community emergency response personnel and for the community as a whole is extensive. Hence, it can be legitimately argued that the existing empirical evidence does not reflect adequately the conditions in CSEPP communities. Further models of the decision to warn will have to recognize these differences and utilize empirical evidence adjusted for these unique conditions. For example, emergency exercise data may be used to validate the approach.

To the extent that PADRE is a planning tool solely for community emergency planners, the implied "black-box" approach is perfectly acceptable. If it is not within the community planner's domain to improve decision-making concerning warning, then why should PADRE simulate potential improvements in that process? PADRE is a tool to help planners improve their response systems, but to the extent that the decision to warn is outside their control, PADRE does not need to simulate it. The problem here is two-fold. First, by giving up the comprehensive nature of the system, the opportunity to discover where and how things may go wrong may also be lost. And second, facilities and communities have rarely, proven to be isolated from each other in chemical emergencies. Although the CSEPP may be a potential candidate for a clean partition between facility and community responsibilities, it remains to be seen that true independence can be achieved, let alone, whether or not it should be saught. None of this suggests that the single parameter options should be removed from PADRE, this approach is often required to examine specific cases in sufficient detail to be useful. Rather it suggests that having options to include distributional qualities for some analyses maintains the probabilistic nature of the underlying knowledge base, which allows the planner to examine the expected exposure associated with a distribution of decision-making times. Another step in this direction would simply use the available distributional information as help information for users. In any event the current, minute-warning-system-activated approach should not be eliminated; however, the inclusion of a distributional approach would maintain consistency with the probabilistic, and the comprehensive nature of the underlying theoretical model. This suggests that future versions of PADRE should consider adding a distributional or probabilistic approach to the decision to warn. This improvement would allow existing empirical evidence to be incorporated into planning decisions.

To repeat, the overall level of agreement here far exceeds the disagreement. One area of considerable agreement is the most logical and appropriate next steps. There is strong concurrence that the present version of PADRE is consistent with the underlying theory. There is strong concurrence that very little (if anything) needs to be done to the present version to maintain that consistency in regard to the decision to warn element of the model. There also seems to be strong concurrence that focusing the community planner's attention on the activities they are most able to influence is adequate and sufficient to the existing planning activities. There also seems to be agreement that a more sophisticated treatment of the decision to warn in chemical events is a candidate for future improvement to PADRE.

Appendix I: Asymtotic Public Response Curves

by George O. Rogers

The Problem:

The estimates of public response in PADRE fit the data very well in terms of variance explained, but overestimate the public response in critical periods. PADRE attempts to empirically generalize from four empirical case studies by representing the cummulative response from each chemical event in terms of a simple model. The basic form of the equation is:

$$PR[t(m)] = 1/\{1+EXP[-a*[t(m)-b]]\}$$

Where PR is the portion of the public responding at time t, t(m) represents the minutes since the receipt of the warning message, a is the slope of the function, and b is the minute at which 50 percent of the population is responding, or midpoint. Table 1 presents the current model parameters.

Table I-1. Summary of PADRE parameters for public response.

	Nanticoke	Confluence	Pittsburgh	Mississauga
PADRE				
Midpoint	30	15	28	29
Slope	0.06	0.11	0.06	0.05

Figures 1 - 4 present the empirical and PADRE estimates of public response in Nanticoke, Confluence, Pittsburgh and Mississauga, respectively. Table 2 presents the simple regression, goodness of fit, and difference statistics for the relationship between the resulting model and the empirically observed public response crurves. The estimates clearly present an excellent fit to the data in terms of overall variance explained, explaining between 96.7 and 98.8 percent of the variance. Unfortunately all the estimates overestimate the public response after the first hour, and three of the four cases are overestimated in the first 15 minutes. Clearly the estimates for Mississauga and Pittsburgh do not approach the same asymptote as the empirical curves (cf. Figure 3 & 4). The estimates for Nanticoke result in a smaller asymptotic difference, and even the estimates for Confluence result in a small difference in the asymptote.

Figure I1. Comparison of empirical and PADRE estimates of public response -- Nanticoke

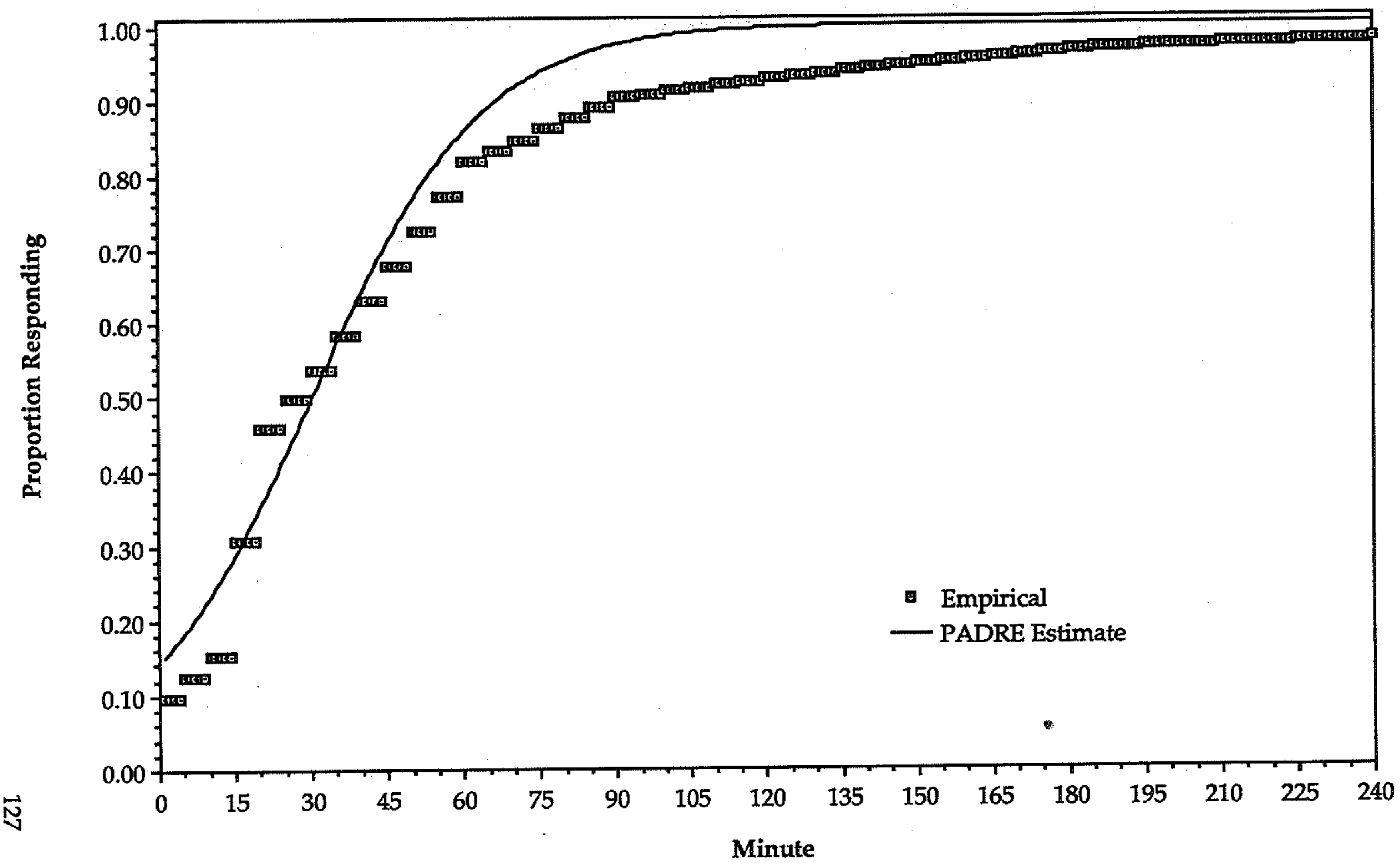


Figure I2. Comparison of empirical and PADRE estimates of public response -- Confluence

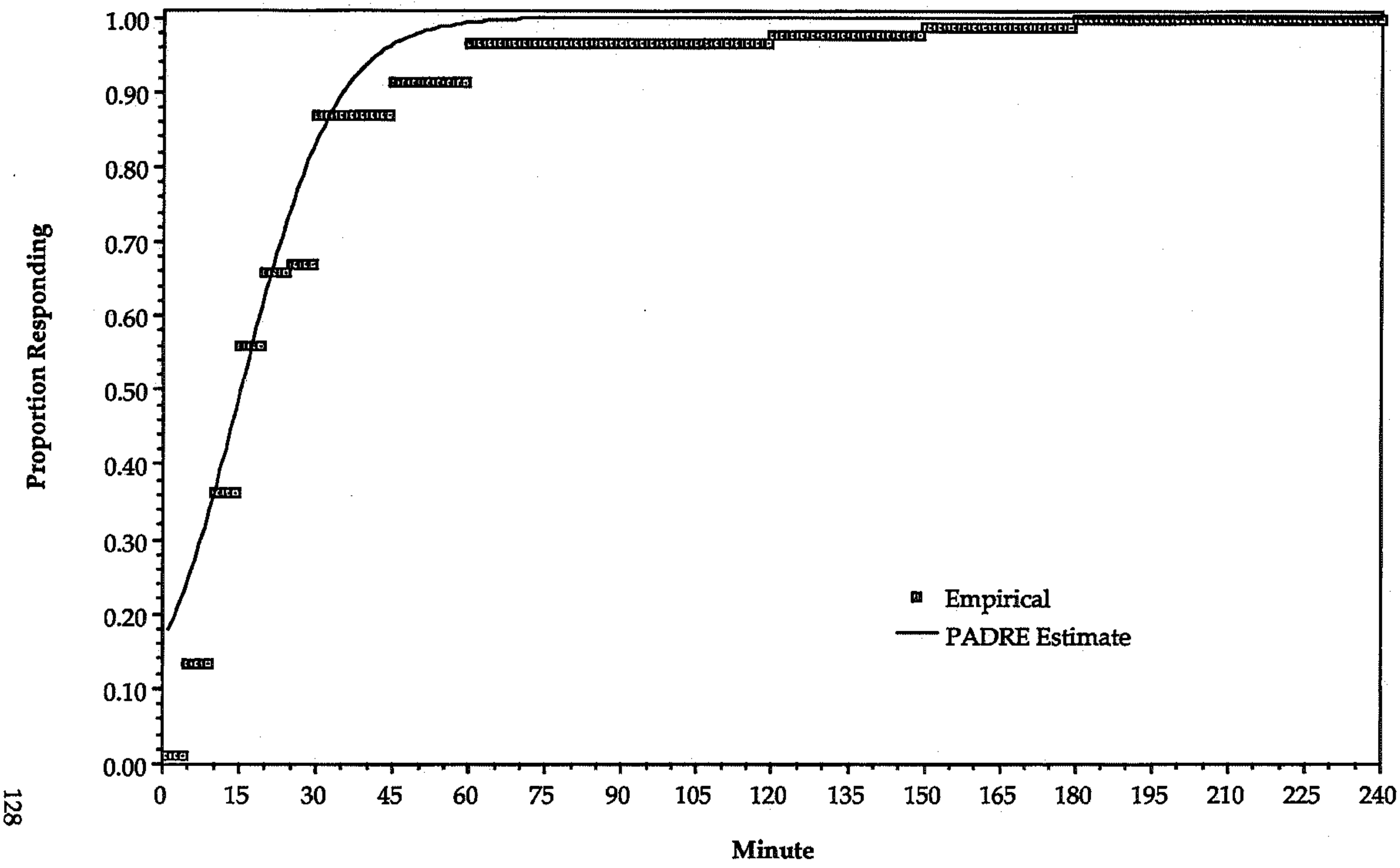


Figure I3. Comparison of empirical and PADRE estimates of public response -- Pittsburgh

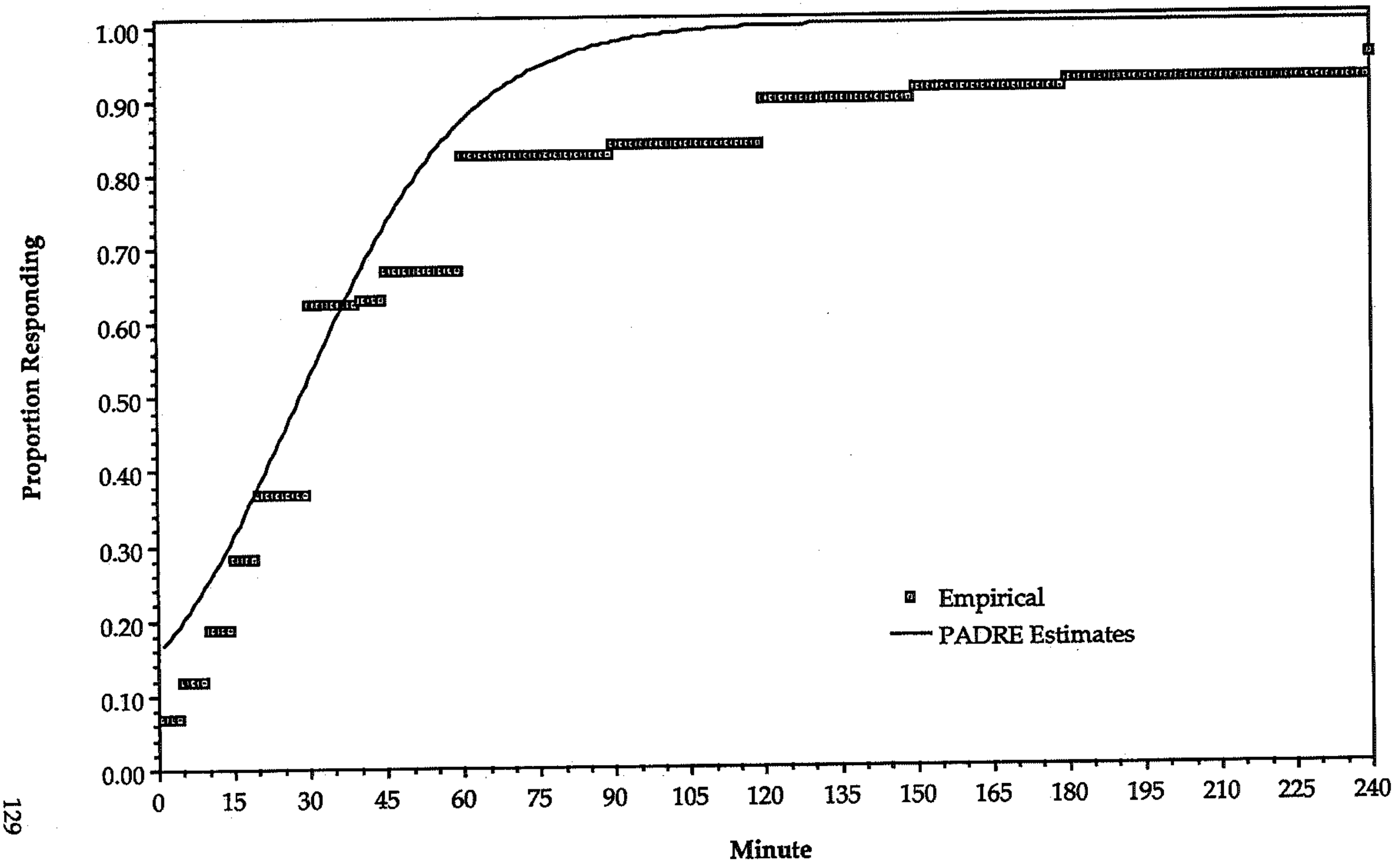


Figure I4. Comparison of empirical and PADRE estimates of public response – Mississauga

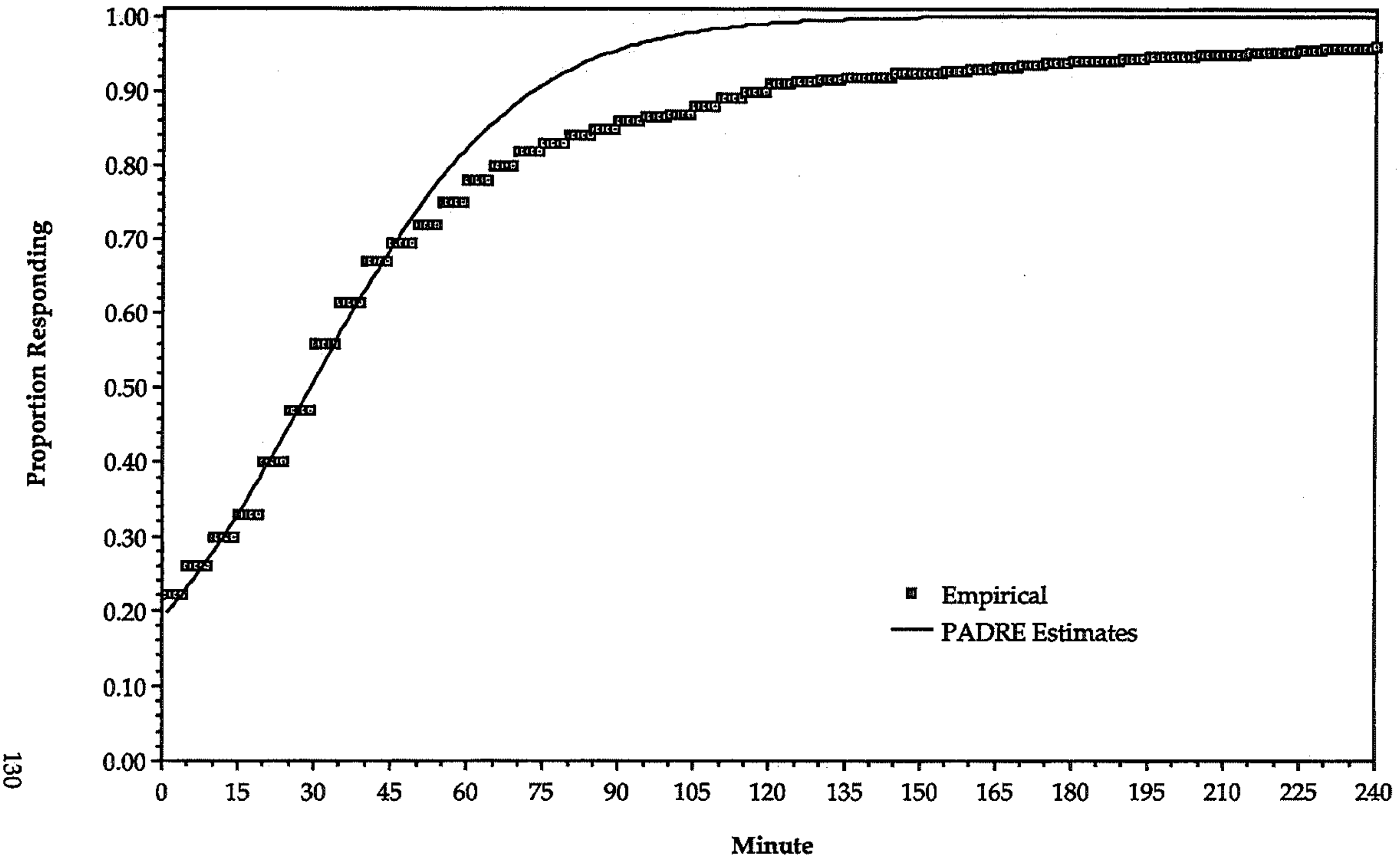


Table I-2. Summary of relationship between existing PADRE estimates and empirical public response curves.

	Nanticoke	Confluence	Pittsburgh	Mississauga
Regression				
R^2	0.9809	0.9758	0.9670	0.9875
Constant	-0.0135	-0.1494	-0.0490	0.0466
Slope	0.9659	1.1349	0.9519	0.8829
Difference				
Average diff	0.0452	0.0297	0.0937	0.0549
Max diff (hr 1)	0.1259	0.2187	0.1963	0.0676
Max diff (hr > 1)	0.0878	0.0330	0.1578	0.1070
Min diff (hr 1)	-0.1057	-0.0600	-0.0950	-0.0475
Min diff (hr > 1)	0.0210	0.0000	0.0440	0.0400

Criteria for Selection

An alternative empirical model of public response will only be recommended if each of the following four criteria are met. The criteria are ranked in terms of importance.

1. Goodness of fit --Any alternatives should explain as much (or more) variance as the existing estimates.

2. Conservative estimates -- Alternatives should underestimate rather than overestimate public response.

3. Parameter efficiency -- Alternatives should use no more than three parameters; the current model uses two parameters slope and mid-point.

4. User convienence -- The parameters should be easily understood by users, which enables users to make informed selection of public response parameters.

Finding an alternative model

Examination of the empirical curves indicated that no simple scale factor could be uniformly applied to existing PADRE estimates to improve the quality of the model. This occurs because the empirical pattern of response results are different for every case. Confluence starts out with the lowest proportion responding and ends with the highest, and Mississauga starts out with the highest and ends with the lowest. Meanwhile, Nantecoke starts out in the middle becomes the least response and by the end of the time period returns to the middle, and Pittsburgh also starts out in the middle range and variously becomes the lowest proportion responding and ends as the least responsive. The variety of pattern among the case studies indicates that no single adjustment to the existing curves will correct the estimation.

Since all of the curves seemed to overestimate, to various degrees, the public response in the asymptote, an asympotic adjustment was saught that would variously adjust PADRE's estimates to account for these variations. This approach can be used to develop better fitting curves; however, the difficulty arises when the amount of overestimation is not significantly reduced, and an additional parameter, which is not easily explained, is used to achieve these results. Specifically, the current estiamte from PADRE, was adjusted by asymptotic term based on time, so that

$$PR[t(m)]' = 1/(1+EXP[-a*[t(m)-b]])*(1-P^{(t(m)-1)}),$$

where all terms are defined as before and P is probability of response within 90 minutes. This results in models that are asymptotic to one, and PADRE's current estimates are asymptotic to one. Hence, no improvement in overestimation was indicated, even though a third parameter was added to the model.

An alternative approach characterizes the empirical results as a "growth curve" with an asymtotic approach to a limit. This approach uses just two parameters, a rate of response, and an asymtotic limit. The probability of public response is

$$PR_t = PR_{t+1} + A * (L-PR_{t+1}),$$

where PR_t and PR_{t+1} are the probability of public response at time t and t+1 respectively, A is the rate of response, and L is the four hour asymptotic limit on public response. Since the four-hour limt for each of the four empirical cases is a known quantity, only the rate of response had to be determined. The systematic approach started with the slope from PADRE and incrementally changed the rate of response (slope) until the prior result fit better than the current model. A one percent increment was used to establish the point at which the model results were not improved. Then a one-tenth of one percent increment was used to determine if the model could be improved. The resulting parameters are presented in table 3.

Table I-3. Summary of asymptotic parameters for public response.

	Nanticoke	Confluence	Pittsburgh	Mississauga
Rate of response (Slope)	0.023	0.046	0.024	0.024
Limit	0.979	1.000	0.956	0.960

Figure I5. Comparison of empirical and estimates of public response -- Nanticoke

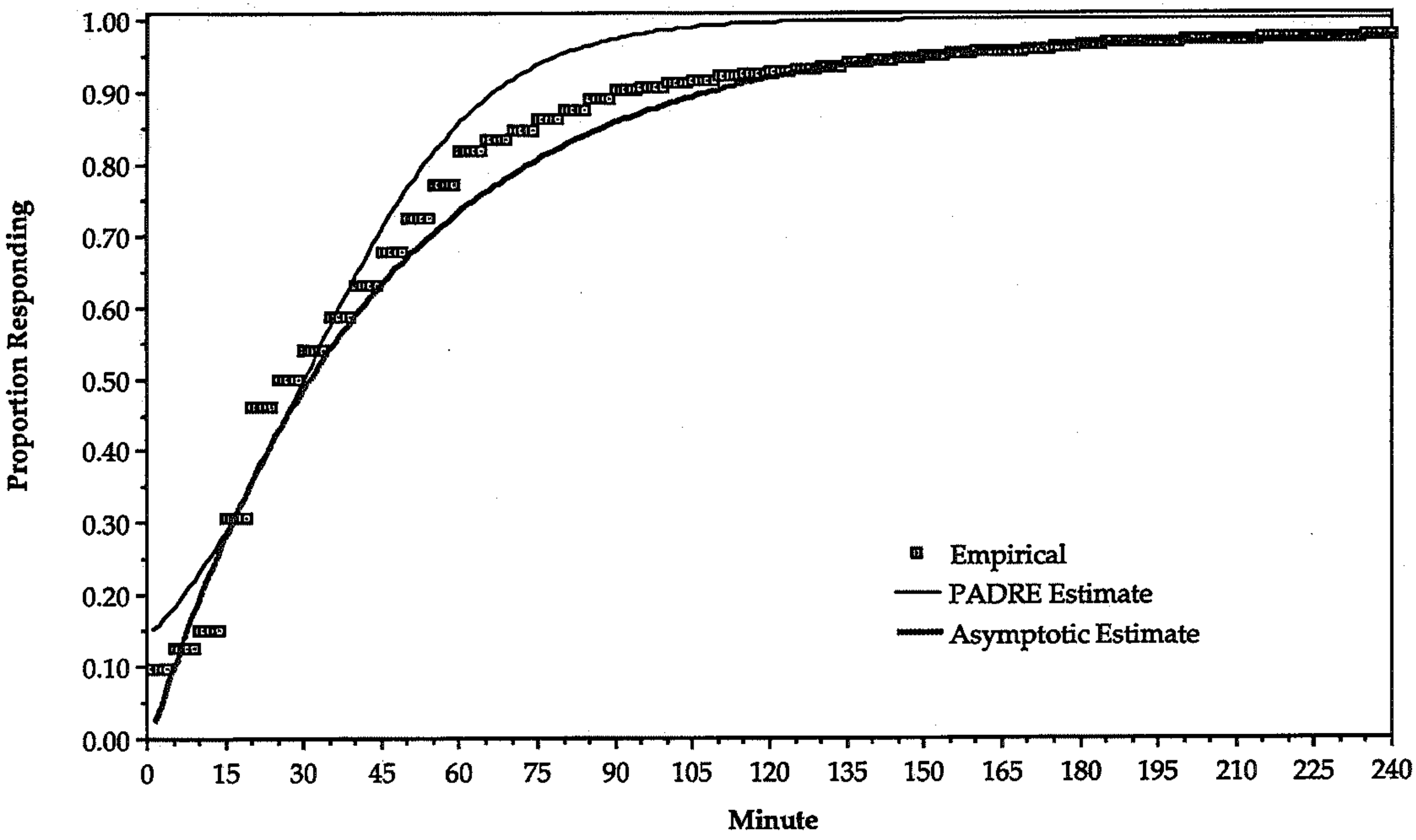


Figure I6. Comparison of empirical and estimates of public response -- Confluence

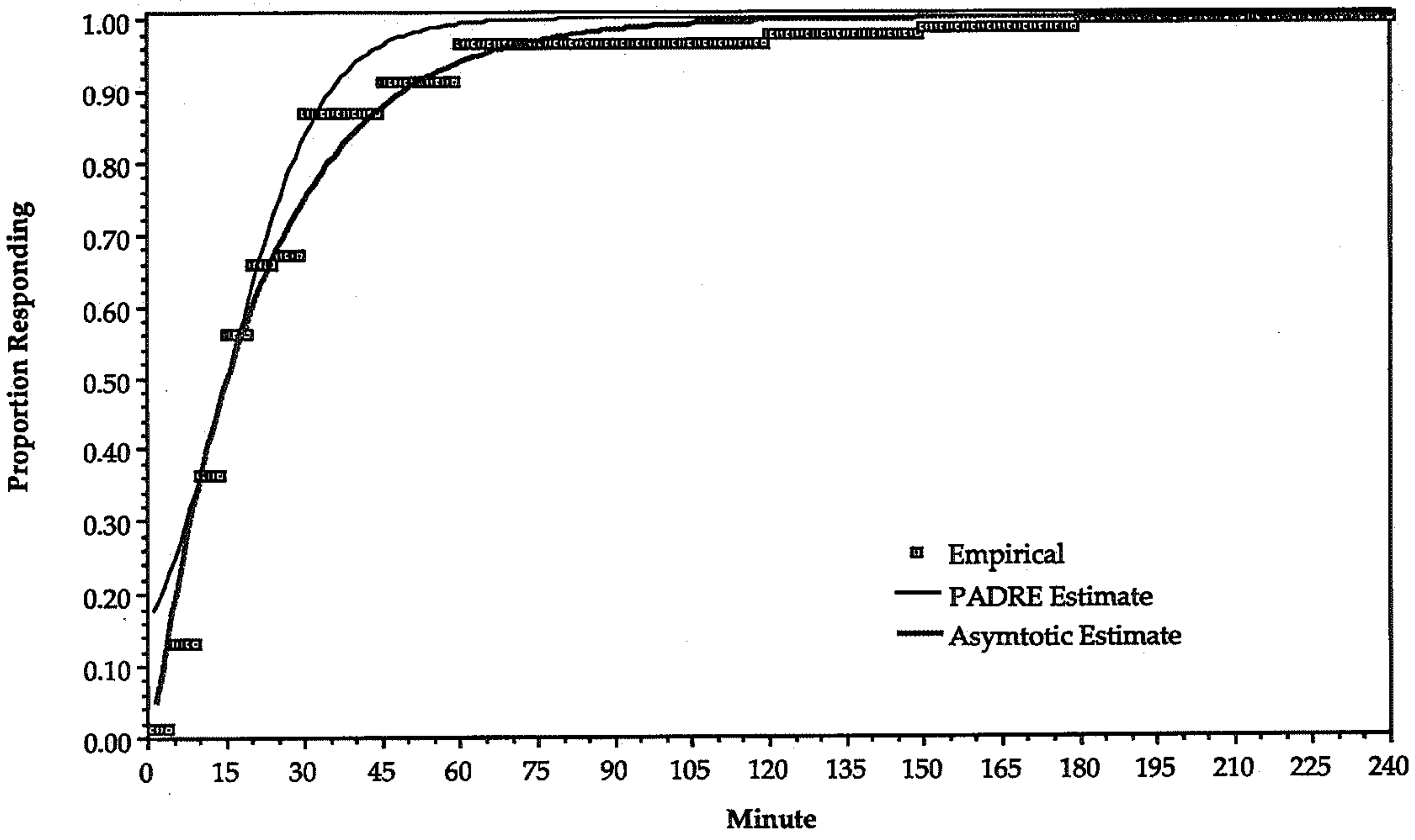


Figure I7. Comparison of empirical and estimates of public response – Pittsburgh

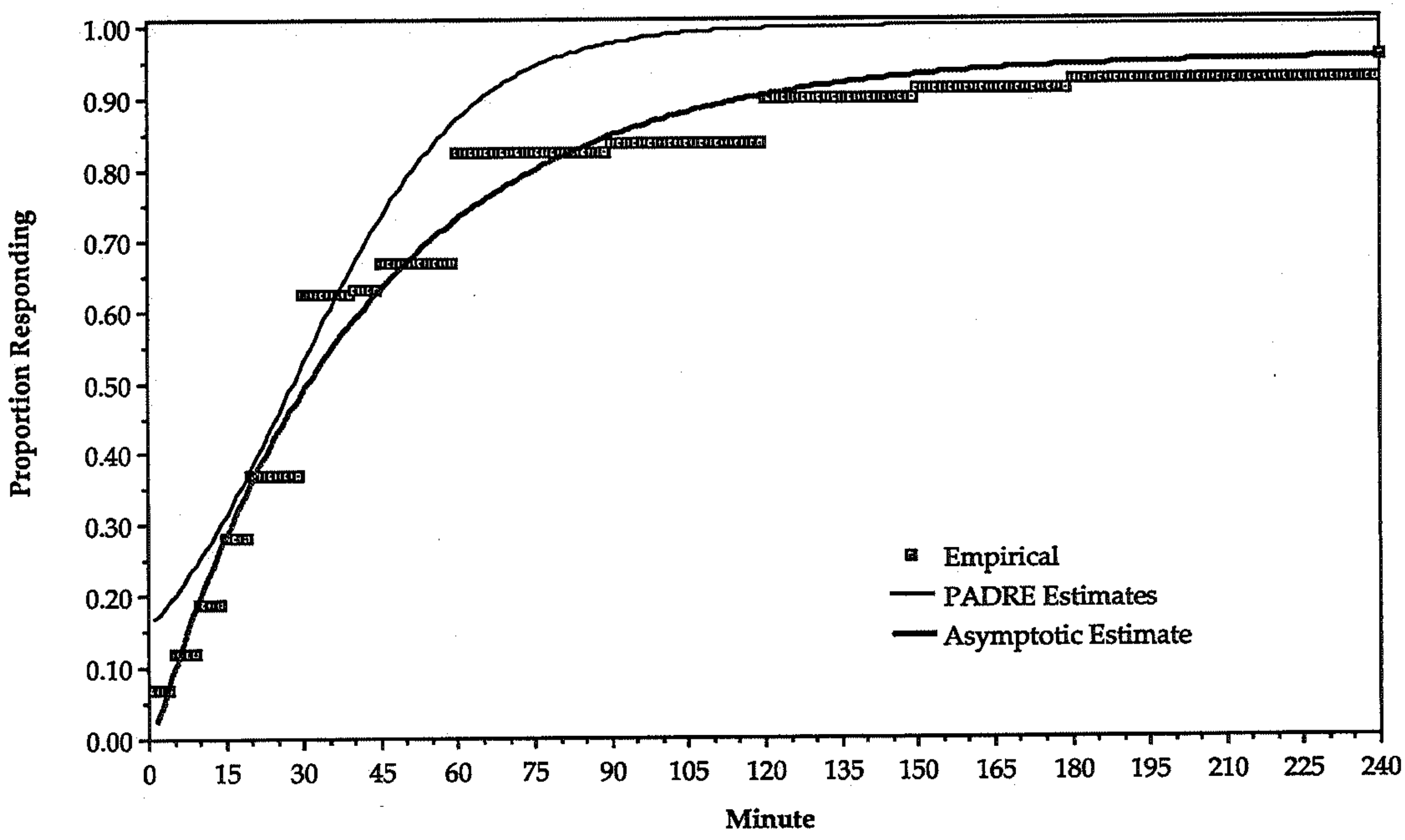


Figure I8. Comparison of empirical and estimates of public response -- Mississauga

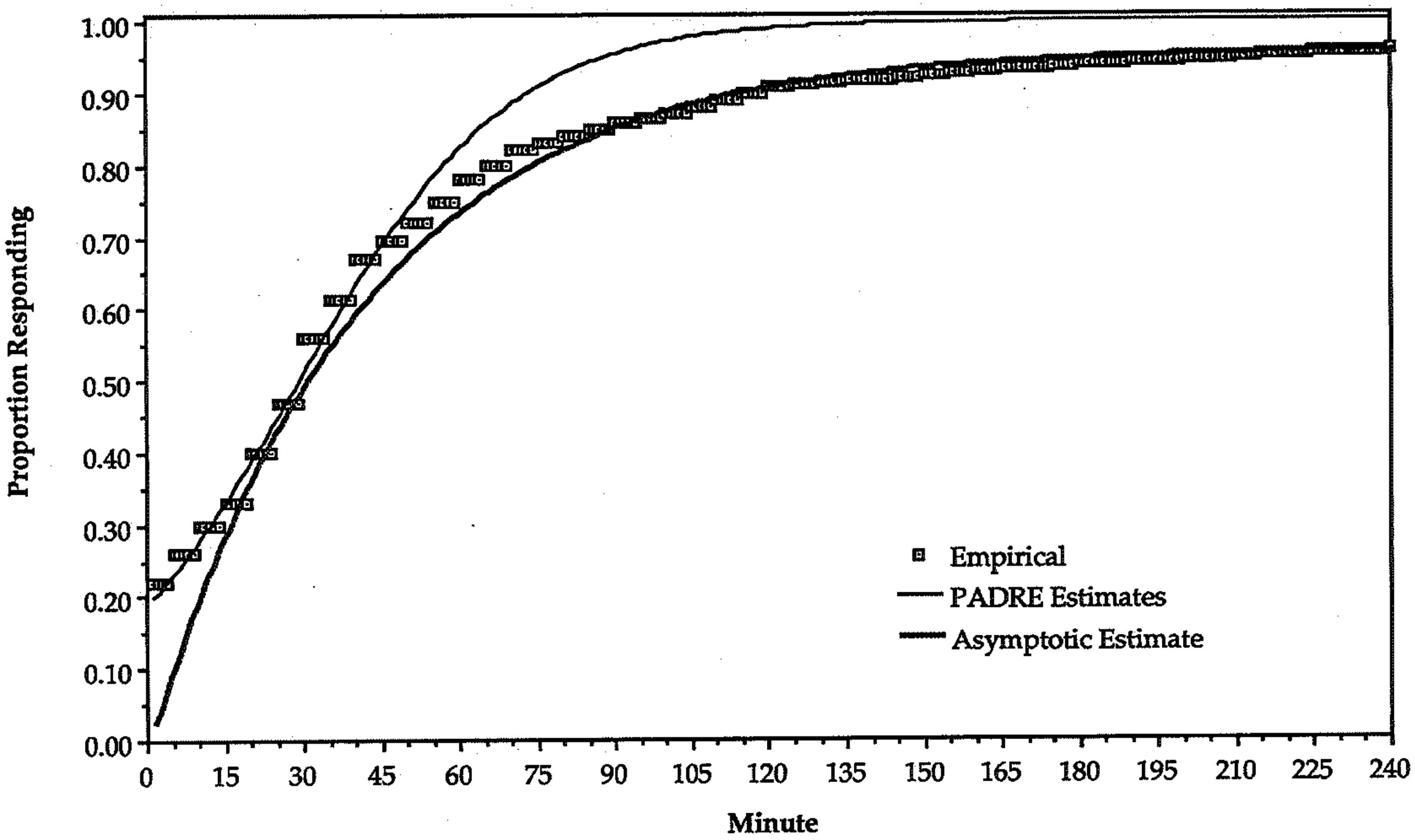


Table I-4. Summary of relationship between asymptotic estimates and empirical public response curves.

	Nanticoke	Confluence	Pittsburgh	Mississauga
Regression				
R^2	0.9854	0.9767	0.9740	0.9904
Slope	0.9959	1.0881	0.9795	0.8909
Constant	0.0155	-0.0919	-0.0003	0.0923
Difference				
Average diff	-0.0145	0.0107	0.0150	-0.0105
Max diff (hr 1)	0.1212	0.2135	0.1144	0.0249
Max Diff (hr > 1)	0.0018	0.0293	0.0649	0.0133
Min diff (hr 1)	-0.0957	-0.1115	-0.1303	-0.1970
Min diff (hr > 1)	-0.0778	-0.0236	-0.0862	-0.0381

Figure 5 thrugh 8 present the empirically observed, PADRE and Asymptotic estimates for Nanticoke, Confluence, Pittsburgh, and Mississauga, respectively. In each case the first three criteria are met. The goodness of fit is improved as indicatd by the improvement in the overall variance explained. In each case the asymptotic estimates are more conservative in that they reduce the amount of overestimation of public response in favor of underestimation. Moreover only two parameters are used to estimate these asymptotic curves. The extent to which these parameters are easier for users to understand cannot be determined with user interaction; however, it seems reasonable to expect users to be comfortable with the concept of the rate of response (or slope) and a four hour limit. The rate of response is at least as easy to estimate as the existing slope parameter, and the four-hour limit may be more easily estimated by users than the current midpoint.

Conclusions and Recommendations

While PADRE's current estimates of public response to warning account for most of the variance in the empirically observered cases, they tend to overestimate public response during certain periods. This is particularly critical early in the response process, but it is also evidenced late in the four-hour period. These overestimates are most clearly observed as the public response curve approaches complete response (i.e., in the asymptote). The purpose of this paper has been to search for new public response estimates that 1) at least maintain the goodness of fit of the current estimates, 2) are conservative in that they reduce the overestimation, even if it is replaced with underestimation, 3) achieve these goals with a limited number of parameters, and 4) use parameters that are easily understood by potential users.

The asymptotic estimates developed herein meet these criteria generally. Hence, there adoption would generalize from the empirical case studies and improve the goodness of fit and conservative nature of the estimates. Moreover, these estimates achieve these goals without additional parameters, and these paramaters seem to be at least as easy for the user to understand as the current parameters. At least one of these asymptotic estimates could be further improved with the addition of a third parameter that characterizes spontaneous public response (e.g., Mississauga), but this would require a third paramter aimed at the proportion of people that are likely to respond prior to being warned. Unfortunately this is a difficult concept for a user to understand; how can people respond to something they do not know about? This tends to occur when the emergency events are protracted and people are able to recognize cues for initiating protective action. Spontaneous public response is least likely in cases where "state-of-the-art" emergency preparedness systems are in place. Hence, the asymptotic estimates selected herein do not include the spontaneous response parameter. The addition, of this term would simply replace the zero starting point assumption with a specified spontaneous response starting point. However, the slopes would have to be re-estimated given the new "intercept." This additional, specification may be best left to an improved theoretical model of pubic response.