1993

Simulation Modeling of Prehospital Trauma Care

Robert L. Wears

University of North Florida

Follow this and additional works at: <https://digitalcommons.unf.edu/etd>

Part of the [Computer Sciences Commons](#)

Suggested Citation

Wears, Robert L., "Simulation Modeling of Prehospital Trauma Care" (1993). *UNF Graduate Theses and Dissertations*. 156.

<https://digitalcommons.unf.edu/etd/156>

This Master's Thesis is brought to you for free and open access by the Student Scholarship at UNF Digital Commons. It has been accepted for inclusion in UNF Graduate Theses and Dissertations by an authorized administrator of UNF Digital Commons. For more information, please contact [Digital Projects](#).

© 1993 All Rights Reserved

SIMULATION MODELING OF PREHOSPITAL TRAUMA CARE

by

Robert L. Wears, MD

A thesis submitted to the
Department of Computer and Information Sciences
in partial fulfillment of the requirements
for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

April 30, 1993

The thesis "Simulation Modeling of Prehospital Trauma Care" submitted by Robert L. Wears in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the thesis committee:

Date

Signature Deleted

4/27/93

Charles N. Winton
Thesis Adviser and Committee Chairperson

Signature Deleted

4/27/93

F. Layne Wallace

Signature Deleted

4/27/93

Susan R. Wallace

Accepted for the Department of Computer and Information Sciences:

Signature Deleted

4/27/93

Robert F. Roggio
Chairperson of the Department

Accepted for the College of Computing Sciences and Engineering:

Signature Deleted

4/27/93

Robert F. Roggio
Dean of the College

Accepted for the University:

Signature Deleted

4/28/93

Charles M. Galloway
Dean of Graduate Studies

Acknowledgement/Dedication

I wish to thank my wife and family for their patience, understanding and support during my second postgraduate education.

Table of Contents

Abstract	xi
Chapter 1 Introduction	1
1.1 Statement of the Problem	1
1.2 Historical Perspective	1
Chapter 2 Description of the System	3
2.1 Definition	3
2.2 System Elements	3
2.2.1 Patients	3
2.2.2 Vehicles	4
2.2.3 Receiving Facilities	4
2.2.4 Transportation Network	5
2.3 System Operation	5
2.3.1 Temporal Sequence	5
2.3.2 Physiologic Sequence	7
2.4 Goals of the Model	7
2.5 Potential Enhancements	8
Chapter 3 Model Design	9
3.1 General Design Issues	9
3.1.1 Simulation Environment	9
3.1.2 Verification and Validation	10
3.1.3 Statistical Issues	10
3.2 Specific Design Issues	12
3.2.1 Patterns of Injury	12

	3.2.2	Transportation Network	13
	3.2.3	Edge Effects	14
	3.2.4	Ambulance Routing	15
	3.2.5	Physiologic Model	15
	3.2.6	Injury Pattern	16
	3.2.7	Critical Outputs	17
Chapter 4		Implementation	18
	4.1	Overview	18
	4.2	Data Structures	18
		4.2.1 Permanent Entities	20
		4.2.2 Temporary Entities	23
	4.3	Procedures	26
		4.3.1 Initialization Procedures	27
		4.3.2 Trace and Reporting Procedures	30
		4.3.3 Modeling Procedures	31
		4.3.4 Flow of Control	37
	4.4	Selection of Input Distributions	39
Chapter 5		Verification and Validation	43
	5.1	Verification	43
		5.1.1 Random variate generators	43
		5.1.2 Static and Dynamic Analysis	45
	5.2	Validation	49
Chapter 6		Demonstrative Experiments	52
	6.1	Triage Policy	52
	6.2	Helicopter Dispatch Policy	55
	6.3	Conclusion	56

6.4	Further Work	56
References		58
Appendix 1	Program Source Code	62
Appendix 2	Data Cross-Reference	119
Appendix 3	Data Files	151
Appendix 4	Sample Output	160
Appendix 5	Fitting Input Distributions	173
Appendix 6	Log of Model Assumptions	183
6.1	Distribution of Accidents	183
6.2	Definitive Care Survival	183
6.3	Private Travel	183
Appendix 7	Log of Improvements and Enhancements . . .	185
7.1	Improvements	185
7.1.1	Transit time	185
7.1.2	Choke points	185
7.1.3	Events	185
7.1.4	End-of-run	186
7.1.5	Memory management	186
7.2	Enhancements	186
7.2.1	Non-trauma patients	186
7.2.2	Injury model	187
7.2.3	Transfers	187
7.2.4	Data editor	187
7.2.5	Graphical output	188
7.2.6	Trace control	188
7.2.7	Interruption	188
7.2.8	Non-regenerative Simulation	189

Appendix 8	Log of Program Bugs	190
8.1	Discrete-continuous Interaction	190
8.2	Pended Accidents	190
Vita		191

Figures

Figure 1.	Logical model of the overall transportation network. (See Figure 3 for node acronymns).	19
Figure 2.	Logical model of the central region of the transportation network. (See 3 for node acronymns).	20
Figure 3.	Spatial distribution of trauma incidents in the study area.	21
Figure 4.	Proportion arrivals in 12 hour periods over one week, corresponding to data in Table 2.	45
Figure 5.	Histogram of 2500 variates from mygamma.f given arguments 1.5, 1.5.	46
Figure 6.	Probability plot of 2500 variates from mygamma.f given arguments 1.5, 1.5. . .	47
Figure 7.	Quantile plot comparing the model's transport times with those provided by Campbell.	51
Figure 8.	Distribution given need to secure.	173
Figure 9.	Probability plot: exponential.	173
Figure 10.	Quantile plot: exponential.	173
Figure 11.	Probability plot: Weibull distribution. . .	174
Figure 12.	Quantile plot: Weibull(6.686, 1.412). . .	174
Figure 13.	Probability plot: lognormal.	174
Figure 14.	Quantile plot: lognormal.	174
Figure 15.	Distribution of time to patient.	175
Figure 16.	Probability plot: exponential.	175
Figure 17.	Quantile plot: exponential.	175

Figure 18.	Distribution of scene treatment time (includes extrication if needed).	176
Figure 19.	Probability plot: Weibull.	176
Figure 20.	Quantile plot: Weibull(11.023, 1.892).	176
Figure 21.	Probability plot: gamma(3).	177
Figure 22.	Quantile plot: gamma(3).	177
Figure 23.	Weibull (dashed line) and gamma pdf's.	177
Figure 24.	Distribution of time to hospital.	178
Figure 25.	Probability plot: Weibull.	178
Figure 26.	Quantile plot: Weibull(9.974, 2.242).	178
Figure 27.	Probability plot: lognormal.	179
Figure 28.	Quantile plot: lognormal.	179
Figure 29.	Probability plot: gamma(3).	179
Figure 30.	Quantile plot: gamma(3).	179
Figure 31.	Distribution of time to release of patient.	180
Figure 32.	Probability plot: Weibull.	180
Figure 33.	Quantile plot: Weibull(2.718, 1.810).	180
Figure 34.	Probability plot: lognormal.	181
Figure 35.	Quantile plot: lognormal.	181
Figure 36.	Probability plot: gamma.	181
Figure 37.	Quantile plot: gamma(1.5).	181
Figure 38.	Probability plot: exponential.	182
Figure 39.	Quantile plot: exponential.	182

Tables

Table 1.	Random variables used in the model.	42
Table 2.	Example data file of mean arrivals by interval for testing the nsp.f routine.	44
Table 3.	Characteristics of ISS scores obtained by the model and those reported by Baker. . . .	46
Table 4.	Portions of trace output demonstrating the manifestation of specific model design items in the implementation.	48
Table 5.	Comparison of outcome estimates produced by the model with those estimated by Jacksonville Fire/Rescue.	49
Table 6.	System performance (mean $\pm$ 95% confidence interval) under different trauma center triage criteria.	53
Table 7.	Convergence points including at least a full seven day cycle.	54
Table 8.	System performance (mean $\pm$ 95% confidence interval) under alternate helicopter dispatch criteria.	55

Abstract

Prehospital emergency care systems are complex and do not necessarily respond predictably to changes in management. A combined discrete-continuous simulation model focusing on trauma care was designed and implemented in SIMSCRIPT II.5 to allow prediction of the systems response to policy changes in terms of its effect on the system and on patient survival.

The utility of the completed model was demonstrated by the results of experiments on triage and helicopter dispatching policies. Experiments on current and two alternate triage policies showed that helicopter utilization is significantly increased by more liberal triage to Level 1 trauma centers, which was expected, but that the waiting time for pending accidents tended to decrease, an unexpected consequence. Experiments on helicopter dispatch policy showed that liberalization of the dispatch policy would have much greater consequences than would changing the triage criteria. Again, this result was unexpected and has received little attention from system planners and administrators, especially with respect to the degree of discussion and controversy surrounding triage criteria.

Chapter 1

Introduction

1.1 Statement of the Problem

Prehospital care of the sick and injured has developed into a complex system in the last 30 years. Much of this development has been "bottom-up," driven by technological factors and the availability heuristic (any available tool will eventually be used). This has eventually led to considerable debate in the medical literature over the appropriate role of several treatment modalities routinely employed in many localities. Furthermore, as resource constraints and other external factors have stressed the system, the need for a systematic overview of the system has become apparent. This project will develop a simulation model of a prehospital trauma care system in order to provide a method by which the effect of modifications to the system can be estimated.

1.2 Historical Perspective

Since prehospital care systems form complex networks of interacting entities that are difficult to work with analytically, simulation has frequently been used as an aid in planning and organizing such systems. The majority of

these simulations have concentrated on relatively static aspects of the system, such as the number and location of responders [Fitzsimmons82, Uyeno84], improvements in response or transport time, etc. [Valenzuela90]. This project will focus more on clinical issues which are more easily modified on a dynamic basis by changing clinical and administrative policies.

Chapter 2

Description of the System

2.1 Definition

The system under consideration is the that portion of the pre-hospital emergency medical care system (EMS) which deals with injury in the seven county service area of northeast Florida and southeast Georgia. The EMS system is obviously impacted by non-traumatic illness as well, so the model must include some representation of their effects, but they will not be the focus of the model.

2.2 System Elements

The system can be decomposed into four fundamental elements: patients, vehicles, receiving facilities, and a transportation network over which vehicles move patients from sites of injury to or between receiving facilities.

2.2.1 Patients. Patients suffer injuries in a particular temporal and spatial distribution. Their occurrence is frequently not independent; for example, most automobile accidents involve two cars and therefore at least two patients. In addition, injuries occur in the two broad, nonexclusive categories of blunt and penetrating. Within these categories, patterns of correlated injuries exist; for

example, brain injury is typically isolated in penetrating trauma, but typically associated with chest and abdominal injuries in blunt. Injuries differ in severity, which affects the probability of survival.

2.2.2 Vehicles. Vehicles in the system are helicopter ambulances, ground ambulances, and private conveyances. Helicopter ambulances are typically few and therefore subject to more stringent dispatching criteria than ground ambulances. The receiving facilities have a degree of control over the destination of ambulances, and receive prior notification of incoming ambulance patients, but benefit from neither with respect to patients arriving by private conveyances. Additionally, ambulance personnel may perform a limited number of therapeutic interventions prior to transporting the patient to a receiving facility. Ground ambulances and private conveyance are constrained to use the transportation network; helicopter ambulances generally travel faster and by line of sight, but are constrained by weather conditions and the need for a safe landing zone.

2.2.3 Receiving Facilities. Receiving facilities in the system are hospitals and other acute care facilities such as clinics or physicians' offices. Hospitals may be classified into Level 1, 2, or 3 trauma centers as defined by Florida statute. Alternatively, they may choose not to participate in the trauma center system; their actual capabilities typically do not change by virtue of this decision.

Receiving facilities will perform initial resuscitation and evaluation of incoming patients, and then transfer them out of the system to definitive care.

2.2.4 Transportation Network. The transportation network consists of existing major roads, highways and bridges. A patient's transport time by ground conveyance is a function of the available path through the transportation network and the time of day. Geographic barriers such as the St. John's River are reflected in the transportation network. Because ambulances are most commonly directly managed by county governments, political boundaries also may affect transportation decisions. For example, in patients with relatively minor injuries, the target receiving facility may be chosen such that the path to it does not involve crossing a county or state line; these considerations are dropped in the face of severe injury.

2.3 System Operation

System operation consists of a temporal sequence of events running in parallel and interacting with a continuous pattern of physiological changes.

2.3.1 Temporal Sequence. A typical cycle begins with an injury-producing episode which generates one or more patients at a particular location and time with a given pattern and severity of injuries. The prehospital system is then activated and an ambulance dispatched to the location,

typically on a proximity basis. The time from injury to arrival on scene is termed "activation time," and will be noted as t_a . Once on scene, EMS personnel may have to locate and/or extricate patients, and may perform some therapeutic services such as starting intravenous fluids, endotracheal intubation, etc. These maneuvers typically will extend the "on scene time" (t_s). Their efficacy is a matter of some debate and could be an item of study in the simulation model.

Once extrication, initial assessment, and initial therapy (if any) have been performed, the patient is transported to a receiving facility in "transport time" (t_t). The means of choosing a receiving facility (e.g., nearest hospital, nearest hospital of a given level, etc.) has also provoked considerable debate, and will be examined in the simulation.

The receiving facility will perform initial resuscitation and evaluation and will then deliver the patient to definitive care (e.g., the operating room, admitted to the hospital, etc.) after "resuscitation time" (t_r) and some additional waiting time (t_x). Definitive care is considered to be outside the system. In some cases, the receiving facility may transfer the patient to another facility, repeating the transport and resuscitation stages of the cycle.

2.3.2 Physiologic Sequence. During this process, the patient's physiological state will change depending on his injuries and the therapy received. Some patients will die before being delivered to definitive care; for those that do not, their probability of survival will be estimated from their injuries and their physiological state at the time of exit from the system [Wears90, Champion91]. Based on their major immediate physiologic effects, injuries can be categorized into three large groups: those producing blood loss; those interfering with respiratory exchange; and those affecting the central nervous system. The physiologic state in each of these deteriorates over time without intervention. Indirect evidence of the severity of injury in these categories is combined into a "trauma score" which is used by EMS personnel to make therapeutic and transportation decisions.

2.4 Goals of the Model

Any simulation model should be constructed to answer specific questions, rather than just show that a model can be constructed. This model will be designed to estimate the effects of changes in:

- a. Triage criteria that determine the center to which a patient should be routed.
- b. Number of trauma centers of specified level.
- c. Criteria for helicopter transportation vs ground transportation.

d. Divert policy (the circumstances and length of time during which a hospital may divert incoming cases to another facility).

e. Location of trauma centers.

These effects will be measured from two perspectives: from the point of view of the system (numbers of patients received, percent utilization, etc.) and from the point of view of the patient (length of time until definitive care, change in survival probability).

2.5 Potential Enhancements

While not an immediate goal of this project, the potential for enhancement of the model to handle additional questions will be kept in mind as a secondary goal. Such additional questions might include analysis of the system during periods of drastically increased demand and/or reduced capacity, as might occur during a natural or man-made disaster; extension of the model to handle non-traumatic medical conditions. Another secondary goal will be portability to other geographic areas without re-compilation; thus to the extent it is practical, area-specific information will be represented by data elements read in from a file, perhaps in a pre-computing step, rather than directly embedded in the program code.

Chapter 3

Model Design

3.1 General Design Issues

General design issues for this project are those common to virtually all simulation models: selection of a simulation environment and the appropriate level of detail, verification of the implementation, validation of model, and the design and analysis of appropriate experiments.

3.1.1 Simulation Environment. The model was implemented in SIMSCRIPT II.5 (CACI Products, La Jolla, CA) for several reasons. SIMSCRIPT is available on a large number of computer systems and has wide general acceptance as a simulation language, thereby facilitating the potential portability of the model. The EMS model proper lends itself easily to discrete simulation, while the physiologic model is more naturally thought of as continuous; SIMSCRIPT provides support for simultaneous continuous and discrete simulation, thus facilitating modeling the interaction between these two components. And finally, local expertise and experience with SIMSCRIPT was available.

3.1.2 Verification and Validation. Separate verification runs checking aspects of the model's logic have been performed and compared to specific test cases derived from available Trauma Registry data. Many of these verification runs were initially performed at the module level so that the desired (true) behavior of the model can be more easily predicted. An activity trace is produced by the model to aid in verification and validation.

The model was validated by checking its output against aggregate data on injury types, patterns of transportation and survival using published data and University Medical Center's local trauma registry. It is unfortunately the case that detailed data on the overall operation of the prehospital care system are not maintained; a modified Turing test may assist in further model validation. The current level of validation of the model is not considered sufficiently definitive for the model to be used in establishing policy. Further validation will require explicit collection of data from the system for comparison to model output.

3.1.3 Statistical Issues. Care has been taken to maintain synchronization of the random number streams when considering policy alternatives; this reduces the variance of the difference between policy alternatives, yielding an increase in statistical power and perhaps a reduction in computing time.

The system under study does not possess well-defined starting and ending times. However, it is the case that the system as defined here does empty out from time to time¹. Therefore, no warm-up period to eliminate the effect of start-up transients was used. Instead, the model is started empty and idle, and the regenerative method will be used to determine run lengths; i.e., a run will be ended when the system returns to the empty and idle state. It should be noted that this method of experimental design might not be desired when the goal is determining system performance under overload (mass casualty) situations; however, only the method of experimentation, not the actual model, would have to be changed.

The primary goal of the model is effect estimation, not hypothesis testing. Statistical testing of the differences between model outputs under differing policies is complicated by the use of the regenerative method, since it cannot be guaranteed that parallel runs will always be directly comparable, even though every random component for each patient is guaranteed to be comparable. For example, individual runs might not necessarily have the same numbers of patients; in general, parallel runs will diverge and reconverge at unpredictable points. A naive direct

1 This does not make it a terminating simulation, because even though the system is empty of patients, the ending value of time for the first run is the beginning value of time for the second, and the time until the next accident is dependent on the current time [Law91].

comparison of alternatives as if they were independent will typically overestimate the variance of the difference in effect. To compare the alternatives properly, summary measures must be calculated at a point where the model has reconverged under each alternative.

3.2 Specific Design Issues.

Certain problems peculiar to this project arose in the development of the model, and were dealt with as follows.

3.2.1 Patterns of Injury. The spatial pattern of injury was assumed to be roughly proportional to population density. This has been shown to be the case in at least one major city [Zachariah92]. Zachariah also showed that the distribution of types of accidents (e.g., assault, auto accident, gun-shot wound, etc.) was to a large extent invariant across time and space; therefore these variables were assumed to be constant in the model.

The temporal pattern of injury was modeled by a non-stationary batch Poisson process, using the method of Çinlar [Çinlar75]. Raw data kindly provided by Zachariah (personal communication) was used to estimate the diurnal pattern of injury occurrence. Variation across days of the week was obtained from Baker92, and the two items combined to produce the weekly cycle of injury incidence used in the model.

3.2.2 Transportation Network. The geographic area of interest was represented at a higher level than blocks or map coordinates by modeling the area as a digraph. Nodes in this graph represent certain critical areas, such as: neighborhoods or fire-rescue service areas from which requests for care arise; choke points -- areas such as bridges which transporters must traverse en route to their destination; and receivers, typically hospitals categorized according to Florida's trauma statute. Arcs in the digraph were assigned weights representing transport time across that arc; these weights may vary with time of day. While some information on average transport times is available from the Fire-Rescue system, information about the distribution of transport times is not. However, Campbell [Campbell92] has published detailed summary results of a variety of pre-hospital time intervals, and kindly agreed to provide his raw data for use in the project (personal communication). Therefore, distributions were fit to Campbell's data using quantile and probability plots, or occasionally using the method of moments.

Since there are extensive and highly functional mutual assistance agreements among the political jurisdictions in the study area, political boundaries have not been explicitly represented in this model. It would be possible, if desired, to represent political boundaries by placing an empirical penalty function on the pertinent arcs; such a

penalty function should be greater for minor injuries and zero for major injuries.

3.2.3 Edge Effects. Only a finite area will be simulated, but resources located near the boundary of the simulated area might be called to service events occurring beyond the boundary; similarly, injuries occurring within the boundary might be managed at hospitals outside the boundary.

Carter⁷⁴ handled this problem by simulating those events at a lower level of detail. However, this merely moves the problem further away, although at a smaller cost than simply enlarging the simulated area. In the system under consideration here, the boundaries tend to fall at "watershed" lines, where events are rare, and very little boundary crossing occurs. For example, it is common for ambulances in St. John's county to respond to calls in Duval county or to transport patients into Duval county. It is very uncommon that they do so with respect to Flagler county, because of the population densities and pre-existing referral patterns. Therefore we will neglect edge effects in this model (although this assumption might be subjected to sensitivity analysis), save for judiciously choosing the boundaries of the simulated area to keep such effects to a minimum. In the current model, the Keystone Heights area of Clay County was removed from consideration, since the flow of referral in that area tends to move towards Stark and Gainesville, i.e., away from the center of the study system.

For similar reasons, only the Kings Bay area in Georgia was included in the model.

3.2.4 Ambulance Routing. Average node to node times within the transportation network routes are precomputed and stored prior to a simulation run. These times are used to generate ambulance call lists for each node, and hospital destination lists for each node, and the basis of shortest expected travel time. The call and dispatch lists are saved in a file that can be edited to reflect special circumstances. Helicopter ambulances are assumed to be callable to any locations, and to alternate calls. The choice of helicopter vs ground ambulance is based on Trauma Score and distance by current policy, and will be the subject of experimentation.

3.2.5 Physiologic Model. Each patient will be represented as a distinct entity within the model, as will resources such as ambulances, helicopters, and hospitals. A limited set of physiologic variables will be modeled for each patient; however, since detailed physiological modelling [Mazzoni88] is computationally intensive, this information will be kept to the minimum necessary to assess probability of survival at different times.

The model of hemorrhage developed by Wears and Winton [Wears90] will be adapted for use in this project. This model can be easily extended to accommodate respiratory exchange as well. Direct CNS injury seems to be a distinct

problem [Baxt87], which is synergistic with both hemorrhage and respiratory injury. It will be modeled as a "black box" process, whose main effect is to cause a downward adjustment in the probability of survival.

The three components of the physiologic model will be used to compute the Revised Trauma Score (RTS), [Champion81, Champion91], which, in conjunction with the Injury Severity Score [Baker74] or ISS, has achieved general acceptance in predicting survival. The RTS assigns each component of the physiologic model a value on a 0 to 4 scale. These scores may then be simply summed to form a 0 to 12 scale, but a weighted sum [Champion91] with a maximum total of 7.804 is thought to provide better prediction. A mapping between the hemorrhage component and these scores has already been developed in Wears90.

3.2.6 Injury Pattern. Injuries occur in identifiable patterns which a model should represent in order to achieve face validity. This would ideally require generation of categorical variables having a given correlation pattern. While many simulation models have assumed independence of variables, apparently successfully, there are several instances [Law91] in which it has been shown that the failure to model correlation between variables substantially affected the results. Devroye [Devroye86] offers several plausible approaches towards the general problem of generating correlated random variates, although he does not

specifically address this particular situation. Alternative approaches have been suggested by Johnson [Johnson87].

Unfortunately, the covariance structure of injury patterns has yet to be described quantitatively. Therefore, it was assumed that blunt and penetrating injuries had the same ISS distribution. Injuries were then modeled by assigning an ISS value (drawn from a scaled beta distribution fit to data from MacKenzie86], partitioning the total ISS among the three major categories of physiologic derangement as suggested in Baxt87 and MacKenzie86, and mapping those components to either direct physiologic variables (e.g., blood pressure) or to RTS components. The Revised Trauma Score values thus computed were validated by comparing their distribution to the distribution of TS reported by Champion81 and Morris86, with good agreement.

3.2.7 Critical Outputs. Certain critical variables were used as the basis of comparison between policy alternatives. These included the dynamic proportion of utilization of trauma centers at each level. Since trauma centers typically must maintain excess capacity, an alternative measure of utilization, the proportion of time the center is at or over capacity, will also be tracked. Other important outcome measures include the total time in the system, the mortality in each phase prior to definitive care, and the overall probability of survival following definitive care.

Chapter 4

Implementation

4.1 Overview

The model's realization in SIMSCRIPT is provided in detail in the Appendices. Appendix 1 contains the program code and Appendix 3 the data files used to instantiate the model. This chapter provides an overview of the entire implementation. The geographic area selected was modelled as a digraph as illustrated in Figure 1 and Figure 2. (Routes in these Figures are shown with single lines for clarity only; inspection of the data files in Appendix 3 will confirm the implementation as a digraph). The spatial distribution of trauma incidents used in the model is illustrated in Figure 3, and roughly corresponds to population density in the target area. The remainder of the implementation can be divided into two major sections; data structures and procedures.

4.2 Data Structures

A variety of SIMSCRIPT data structures were used to represent the various model elements. Two general principles were used in representing entities in the model. First, entities having a potential lifespan in the model greater than a typical run length were be represented as

Figure 1. Logical model of the overall transportation network. (See Appendix 3 for node acronyms).

SIMSCRIPT permanent entities, while entities that potentially might "come and go" during the course of a run were represented as SIMSCRIPT temporary entities. Second, no entity should have greater knowledge about itself or about conditions in the system than its would its real-world analog. Application of these principles to the model entities described in 2.2 produced the following set of data structures (lines 39 - 256 in the preamble, Appendix 1.)

4.2.1 Permanent Entities. The following structures are set up by the initialization code and exist throughout the entire simulation.

Figure 2. Logical model of the central region of the transportation network. (See Appendix 3 for node acronyms).

4.2.2 Nodes (lines 43 - 54). Nodes in the transportation network represent areas in which accidents might arise. They are identified by location (latitude and longitude), and have an `edge.set` of arcs representing paths to and from other nodes. Additionally, nodes can own ambulances or hospitals if one is located in a node's area. Each node has a list of ambulances to call for events occurring in its area, and a list of hospitals to which patients will be transported from accidents occurring in the node. Nodes are

Figure 3. Spatial distribution of trauma incidents in the study area.

collected in sets so that nodes containing hospitals or ambulance bases may be easily identified.

4.2.2.1 Paths (lines 56 - 63). Every node - node pair is connected by a path in each direction. The path is held in the pair's route set, and consists of the sequence of arcs to be traversed in proceeding from one node to the other. Each path is associated with a mean transit time and a flight time. Unidirectional paths may be implemented by

assigning them an essentially infinite travel time in the reverse direction.

4.2.2.2 Hospitals (lines 65 - 100). Each hospital in the system is assigned a level according to Florida Trauma Center designation standards. Hospitals also are assigned a capacity, based on the maximum number of active resuscitations they can handle, and a number of flags indicated whether they are allowed to divert, whether or not they have diverted in a given period, etc. Hospitals also track the number of patients they are currently resuscitating, and a variety of other statistical counters. Hospital divert status is represented by membership in a green (no divert) and a red (divert) set.

4.2.2.3 Ambulances (lines 102 - 117). Each ambulance has a type, indicating whether it is a ground or an air (helicopter) ambulance, and a base node. It also maintains a pointer to an ambulance run process (if any) representing an actual run, and has storage for its current location, although this attribute is not always guaranteed to be current. Ambulances belong to a variety of sets to track their activity, the most important being the ready.set. Membership in the ready.set indicates the ambulance is available to be called to an accident. This representation was chosen over SIMSCRIPT's built-in 'resource' entity since ambulances are not entirely interchangeable.

4.2.3 Temporary Entities. The following structures may be created and destroyed as needed throughout the simulation. The procedures (if any) associated with temporary entities are discussed in Section 4.3.

4.2.3.1 Arcs (lines 121 - 132). Arcs representing logical (not necessarily physical) routes of travel are implemented as temporary entities under the supposition that they could, at least in theory, come into and disappear from existence during the course of a simulation. Arc's are unidirectional and are identified by their source and sink nodes, and a weight representing the average travel time in minutes from the center of the source node to the center of the sink node. A choke weight is also available to represent the average additional delay that might be experience at a choke point. Cumulative weights are used in the calculation of best routes from node to node, but are not subsequently used during the simulation. Since an arc may not belong to more than one set of a given type in SIMSCRIPT, but a given arc may be part of many different routes, duplicate arcs are created, at some cost in storage space (see Appendix 7.1.5).

4.2.3.2 Dispatch lists (lines 134 - 143). Two dispatch lists are maintained by each node; a list of call.items (pointers to ambulances) to be called for incidents occurring in that node, and a list of go.items (pointers to hospitals) indicating destination hospitals for incidents occurring in a node.

4.2.3.3 Ambulance runs (lines 148 - 165). An ambulance run is represented by a process that is created when the dispatcher assigns an ambulance to an accident, and ends when the ambulance returns to its base and is back in service. Ambulance runs are divided into two kinds, trauma and medical. An ambulance run is always associated with a particular ambulance, and has attributes for identifying the accident it is serving, the node from which it travels, and the node, hospital, and hospital's level to which it is bound. Two flags are maintained: status, to identify when a run ends in a recall, and helo.coming, to indicate when an ambulance should wait for the helicopter's arrival, even though it would otherwise be ready to travel (i.e., its scene.time is over). Ambulances keep track of the patients they manage on a run by filing them in the amb.patient.set.

4.2.3.4 Patients (lines 167 - 213). Patients are represented as processes created by accidents, and are destroyed when they either die or are transferred to definitive care. A patient's condition may be alive or dead; they move through several phases (e.g., awaiting treatment, scene treatment, transport, etc.), with phase changes being triggered by setting the change.flag. Physiologic information about a patient consists of the hemodynamic components of the Lewis model of hemorrhage [Lewis86] as modified by Wears and Winton [Wears90], extended to account for the impact of respiratory

embarrassment on oxygen delivery. The patient's degree of injury is measured by the ISS. From the ISS and systolic blood pressure (sbp), the components of the revised Champions trauma score, and the score itself can be determined. A functional attribute, cts.f, allows the trauma score to be updated periodically to reflect the patient's changing condition. Finally, patient maintains a variety of time intervals of interest for reporting purposes.

4.2.3.5 Accidents (lines 216 - 235). Accidents are represented as processes that are created by the generator and are destroyed when the last patient associated with an accident is removed from the accident site. Accidents are of two kinds, medical and trauma. In the current model, medical accidents are served by ambulances just as trauma accidents are; they create no patients but do constitute a demand on the system. Trauma accidents may create several patients, and may be blunt or not (i.e., penetrating). Both medical and trauma cases may be placed on a pending list if insufficient ambulances are available to meet their needs.

4.2.3.6 Events (lines 240 - 244). Events were used to handle a hospital's going on and coming off of divert status. A hospital that places itself on divert status frequently must reopen to ambulances after a certain period of time, regardless of its status at that time. The event go.off.red is used to schedule this status change. In

addition, many jurisdictions clear all their divert status once a day; the event `clear.reds` performs this function. Finally, the event `resp.support` is used to model the effect of therapeutic interventions assisting respiration and ventilation occurring in the course of scene treatment or resuscitation.

4.3 Procedures

The model has a natural structure that can be described as a collection of independent but communicating entities. This suggests that an object-oriented approach would have provided the most natural implementation. Since an object-oriented simulation environment was not available, a monitor process was used to handle interprocess communications. A natural monitor, the dispatcher, exists in the real-world system, so this approach meshed nicely with the target model. Interestingly, the monitor function was more easily provided as a procedure, rather than as a SIMSCRIPT entity. Thus the dispatcher is the only major real world entity that has only an implicit representation in the model.

In this implementation, procedures can be divided into three classes: initialization procedures, trace and reporting procedures, and the actual modeling procedures (with their supporting utility procedures). The individual procedures are described here; their relationship and the flow of control among the procedures is described in Section 4.3.4.

4.3.1 Initialization Procedures. The initialization procedures get system information from a set of files and create the data structures outlined in Section 4.2. They are not called again and in principle could be physically separated from the simulation code itself.

4.3.1.1 Main (lines 417 - 498). Main is the fundamental control routine in a SIMSCRIPT program. Main opens the output files and then calls the routine initialize (see 4.3.1.2). At this point, main could begin a loop for each arm of an experiment; this has been left for future implementation if desired. Next, important global variables such as time.v, nsp.tprime and nsp.last.time (used by the non-stationary Poisson routine nsp.f) are initialized to zero, and the run loop is entered. This loop resets all the runwise totals and schedules the next time for clearing hospital divert status, and then activates the generator process to start the simulation. Once a run is finished, runwise statistics are calculated and the run.report routine is called, and then the loop is re-executed until the requested number of runs has been obtained. The routine final.report is then called and the program terminates.

4.3.1.2 Initialize (lines 2333 - 2365). Initialize gets data to characterize this instantiation of the model. It calls a series of initialization routines (get.table, get.sim, get.net, get.hosp, get.ems, get.accs) that read information from datafiles into model variables in a

straightforward manner. In addition to simply reading in data on the transportation network, the routine `get.net` creates the digraph of nodes and arcs, and calculates mean transit times to all nodes in the system by calling the function `best.route` (see 4.3.1.5).

The ambulance call list and hospital preference list are either read from datafiles, if they exist, or are constructed (and written to datafiles for editing or future use) using default dispatching rules if they do not. This process is controlled by the routine `get.list` (lines 1853 - 1873), which is driven by its subprogram variable (SIMSCRIPT's term for pointer to function) arguments. The actual construction of the lists is performed by the routines `build.call.list` (see 4.3.1.3) and `build.hosp.list` (see 4.3.1.4). Before returning, the `initialize` calls the routine `print.net` to output the data it has read for verification of the model's initialization.

4.3.1.3 `Build.call.list` (lines 884 - 972). `Build.call.list` uses the internode transit times along the best path to construct for each node a list of ambulances to be called for incidents in that node, based on least travel time. The default rules governing ambulance selection are: helicopter ambulances will travel to all nodes; every node will have a minimum of `min.amb` ambulances on its list; ambulances are ranked on the list in order of closest travel time; and that travel times that are within a given proportion of each

other (stored in the global variable atol) are presumed to be sufficiently equal that all such units will be placed on the list, even if the minimum number of ambulances is exceeded. This is necessary to handle ties in travel time, and to prevent the list from being unreasonably limited by small differences in travel time.

4.3.1.4 Build.hosp.list (lines 975 - 1069).

Build.hosp.list constructs for each node a list of hospitals to which victims in that node will be sent. It is functionally similar to build.call.list (above), but handles the additional complication of maintaining a list for each trauma center level. The default rules here are analogous to those for build.call.list, except that every node must have at least one Level 1 center and at least one Level 2 center on its list. Again, as many hospitals whose travel times are within a given proportion (htol) of each other may be put on a list.

4.3.1.5 Best.route (lines 752 - 828).

The function best.route is given a source and a destination node as arguments and returns the shortest possible travel time between nodes using Prim's algorithm. If the path involves an arc identified as a choke point, the path is penalized by adding the choke point's weight (representing the mean additional travel time) to the regular weight. Once the best route has been determined, the routine build.route is called to file the ordered list of arcs in the route set

owned by the from.node, to.node compound entity. Finally, since the calculated travel time has been based on inter-node transit times over arterial highways, the function adj.time.f is called to adjust for the time it typically takes to move from secondary and tertiary roads to major highways and back again, and this adjusted time is returned to the caller.

4.3.2 Trace and Reporting Procedures. A collection of procedures named tr.XXX provides the majority of the trace output (lines 3087 - 3344). These procedures are triggered by filing or removing an entity from a set (see lines 383 - 411 in the preamble). They do not provide any service in the model other than the trace, and so in theory could be commented out in a final, validated version. However, since trace output can be redirected to a file (or to the NUL device), and since a trace is extremely helpful in debugging, there should be no reason to remove these functions.

In addition to the trace functions there are three other reporting functions: run.report, final.report, and pt.report. Run.report (lines 3006 - 3084) is called after the completion of a run and writes summary information on that run to two files: a text file called summary.res and a formatted data file named run.res. Summary.res contains easily read summary information for a quick impression of the model's output, while run.res contains detailed

information in a record format where each run is a line, and each field in that line is a numeric data item such as the run number, duration, midpoint, number of accidents, etc. This format is easily loaded by spreadsheets and statistical software, so that more sophisticated analysis can be performed.

4.3.3 Modeling Procedures. The following section explains the flow of control in the model, followed by documentation of the procedures involved in the actual execution of the instantiated model.

4.3.3.1 Generator (lines 1591 - 1625). If this instance of the generator process is for the first run of a series, the process calls the function `nsp.f` to determine the time until the next event occurs. It waits until that event, and then enters a loop where it creates, initializes and activates an accident process; generates the next inter-event time and waits for that event; and then just prior to starting the next event, checks to see if the run termination criteria have been met. If they have, it cancels any scheduled `clear.reds` events (because this might be the last run), but saves the time remaining (the `time.a` attribute of `clear.reds`) so that a `clear.reds` event can be scheduled at the proper time in the following run (if any).

If this instance of generator is not the first run in a series, the previous run's value of `time.v` is retained, and the run will begin by activating an accident process.

4.3.3.2 Accident (lines 501 - 537). The attributes of the process accident will already have been initialized by the procedure `init.accident` (lines 2204 - 2258). These attributes include the location, type of event (trauma or medical), number of victims, and type of trauma (blunt or penetrating) if traumatic. `Init.accident` also initializes (by calling `init.pt` (lines 2261 - 2330) and activates any patients that it has created. Once the accident process is activated, it uses an empiric function based on average severity to compute a probability that the victims will find their own way to the nearest hospital (greater severity implies lower probability of private transport). This function is based on expressions of probability by various domain experts; no data could be found on this topic.

If private travel is chosen, the routine `pvt.travel` (see 4.3.3.7) is called. If not, the dispatcher is sent an ambulance request after a lag time. This request is updated with more accurate information after another lag. Once the dispatcher has been fully notified, the accident process suspends itself. It has no further work to do, but must remain in existence until all patients have been picked up by an ambulance or have otherwise left the scene, since the only way to keep from losing track of a patient is to keep

it filed in the acc.patient.set until an ambulance assumes responsibility for it.

4.3.3.3 Ambulance.run (lines 561 - 728). The ambulance run process is complex, since it must represent two different ambulances and the interaction between them. The process does some initialization and calls the utility routine travel (lines 3367 - 3377) to model travel to the accident site. It may be interrupted by the dispatcher to be recalled during this time. Once at the scene, the process may take one of two branches (lines 602 - 628 and lines 628 - 671) depending on whether the ambulance is an air or ground unit. The branches are largely similar except for their provision for inter-ambulance interaction; an air ambulance takes a patient from a ground ambulance and leaves, while a ground ambulance must consider if it needs an air ambulance, wait for one to arrive if it has been called (even if the ground unit would have been ready to roll), obtains its patient(s) via the get.patient routine, may give its patient to the helicopter ambulance and then check to see if there are more patients needing care before leaving. Ground ambulances also must handle the medical calls, since these rarely require helicopter transportation. Finally, as each ambulance leaves the scene, it calls the routine check.accident which does housekeeping on the accident process; if all the patients belonging to that accident have been taken care of, it reactivates the

accident so that it may end itself. Ambulance.run then uses the travel routine to model traveling to the hospital and returning to base, notifying the dispatcher of its status at appropriate points along the way.

4.3.3.4 Patient (lines 2646 - 2756). The patient process is used to manage the continuous simulation routines. The design is to enter a 'work continuously' statement, using the function bleed to model the patient's physiologic status, and the function done to determine when this particular phase of the patient's experience is over. The update function is used to make the continuous variables visible to allow for reporting if desired.

The logic used to handle a patient's death requires some explanation. The function living is called periodically to determine if a patient meets the criteria for death. Death before ambulance arrival ends the patient process. In any other phase, death is recorded at the time it occurs, but the patient process continues through the end of the initial resuscitation. This reflects the real world system, in which the initiation of field therapy mandates transport to a hospital and resuscitation, even after the loss of vital signs, before pronouncing a patient dead. This complication is handled by using the routine pass.time (lines 2581 - 2643) to control the continuous simulation statements, and to reexecute the 'work continuously' statement if a patient dies during continuous simulation.

Finally, if the patient survives to be delivered to definitive care, his probability of survival is calculated from his ISS and RTS, using logistic regression coefficients from the MTOS [Champion90].

4.3.3.5 Dispatcher (lines 1238 - 1391). The dispatcher routine serves to coordinate the various processes in the simulation. It consists entirely of a multiway branch (select case) statement. Only one branch is executed each time dispatcher is called. The branches correspond to the various notifications that the real world dispatcher receives, such as request for ambulance or helicopter, updates to previous requests, and reports from ambulances on their status (en route to scene, on scene, at hospital, returning to base, and need assistance). The dispatcher also manages requests that can't be satisfied by filing requesting accidents in a pending set. When an ambulance is placed back in service, either when it returns to its base, or when it is recalled from a run, the dispatcher checks the pending set to see if that ambulance can be dispatched to an accident (if any) on the pending list. Finally, the dispatcher receives notification when the last patient is taken from an accident scene; it can then recall any ambulances still en route, and reactivate the accident so it will terminate.

4.3.3.6 Get.patient (lines 1944 - 1986). The get.patient procedure updates every patients current RTS and ensures that patients are assigned to ambulances in RTS priority. It then chooses a destination level based on the logic previously described, although this is overridden in the case of helicopter ambulances who always transport back to their own base hospitals, unless a higher level of care is required. Get.patient does some housekeeping by moving the patient from the accident to the ambulance's set, setting intravenous fluid starting times and respiratory support times for the patient, and then uses the system's rules for deciding whether to request the helicopter or not. Get.patient only assigns one patient, and so it is called repeatedly to assign multiple patients to an ambulance if their acuity levels are low enough.

4.3.3.7 Pvt.travel (lines 2892 - 2930). The routine pvt.travel handles a patient's transportation to a hospital outside of and unknown to the EMS system. In this setting patients are assumed to go to the nearest hospital, regardless of its ability to handle their injury, and regardless of its divert status. Private transport is also assumed to take longer than EMS transport, although the overall time may be shorter since there is no wait for the ambulance to arrive and no on-scene treatment.

4.3.4 Flow of Control. The flow of control in this implementation, as in the real world system, is complex. The dispatcher function is critical to understanding the implementation, since it serves as a monitor to coordinate communication between the independent entities in the model. Although not specifically implemented as such, entities such as ambulances and patients can be viewed as finite state automata, with the dispatcher functioning to oversee state transitions. A typical sequence is given in the following.

An episode begins when the generator function creates an accident, and initializes it to have some number of patients, which some constellation of injury characteristics. Each patient process is activated and enters a continuous simulation phase that models the physiologic effect of its injuries. After a brief delay, the accident notifies the dispatcher of its existence, location, an approximate number of patients. The dispatcher determines the number of ambulances that should respond, selects particular ambulances from those available, and creates and activates ambulance run processes for each. Finally, the dispatcher files the accident in the pending set if more ambulances are needed, and then exits. The dispatcher function is reinvoked upon arrival of an ambulance at the scene, or after a lag interval, whichever occurs first. At this point, information assumed to be accurate about the number of victims is sent to the

dispatcher, which may in turn recall some of the enroute ambulances, or may dispatch additional units as needed.

At this point, patient and ambulance run processes are executing simultaneously. Once an ambulance run process was worked for its designated travel time, it notifies the dispatcher that it has arrived on scene, and after a brief interval, picks up patients from the accident in order of apparent severity. Once the ambulance run "owns" some patients, it adjusts the patient's characteristics to reflect interventions such as respiratory support and intravenous fluid therapy. It also determines the appropriate destination, based on the level of care needed for the degree of injury. (One might think the dispatcher should do this, but the real world system operates in this manner). The ambulance run may also call for the helicopter ambulance to come to the scene and take over, depending on the severity of injury and distance from the appropriate level of care. The accident process terminates when the last patient is picked up by an ambulance.

After scene care has been rendered, the ambulance run notifies the dispatcher that it is enroute to a hospital. This notification also includes a request for additional help if there is still a disparity between the number of victims remaining and the number of ambulances on scene or enroute to the accident. The dispatcher updates the

ambulance run's set membership appropriately to reflect this new status.

After the hospital transport time has passed, the ambulance run notifies the dispatcher of its arrival at the hospital. The patient(s) are then transferred to the resus.set, and undergo their resuscitative care, after which they are transferred to definitive care and leave the system. After a cleanup time, the dispatcher sends the ambulance back to its base, and upon arrival there, refiles the ambulance in the ready.set and terminates the ambulance.run process. The cycle is now ready to begin anew.

4.4 Selection of Input Distributions

The distributional form of the input random variables was chosen after consideration of both theoretical and practical issues. For example, for those distributions known to be bounded, beta distributions were chosen since they were also bounded, and were then scaled and fit using moment matching or maximum likelihood methods. Similarly, if a distribution was known to be skewed to the right, or nonnegative, candidate distributions were restricted to those having the appropriate general characteristics.

For all distributions for which empirical data was available, the choice among candidate distributions was made by visually assessing probability and quantile plots [Law91], after matching the first two moments (mean and

variance) to the empirical data. For each quantile ordinate, the quantile plot graphs the abscissa of the candidate distribution corresponding to that ordinate against the abscissa of the empiric distribution corresponding to that ordinate. A good fit will thus appear as a straight line, and differences in the tails of the candidate and empirical distributions will show up as deviations from a straight line. In the probability plot, for each abscissa value, the ordinate of the cumulative distribution function of the empiric distribution is plotted against the ordinate of the candidate distribution for the same abscissa value. Again, good fit will appear as a straight line, but differences in the middle of the distributions are magnified and appear as deviations from a straight line. In addition, the slope and intercept of the plots (or the logged plots, for Weibull distributions) can be used to estimate the parameters of the fitted distributions [Wilkinson90]. Several examples of distributions fitted using these techniques are given in Appendix 5.

Finally, although several of the time duration variables were well-approximated by Weibull distributions, the final decision was made to use gamma variates instead, since the parameters of a gamma distribution are simple functions of the mean, which is not the case for the Weibull. This allows more flexibility in using the model, since mean times

can be easily changed in the data file. Inspection of the probability and quantile plots in 5 shows that this is not likely to produce large differences in the output, since gamma distributions also fit the data very well. A listing of all the random variables used in the model, the distributions and arguments chosen to generate them and the random number streams assigned is provided in Table 1.

module	variable	distribution	arguments	stream
accident	notification lag	exponential.f	alarm.lag	15
accident	pvt vs ambulance	random.f	n/a	
accident	update lag	exponential.f	info.lag	3
ambulance.run	scene time	mygamma.f	t.on.scene, 3	4
"	time to patient	exponential.f	t.to.pt	5
"	cleanup time	exponential.f	3	6
"	secure time	lognormal.f	m.secure,	
			s.secure	18
"	need secure time?	random.f	n/a	19
"	deliver time	lognormal.f	m.deliver,	
			s.deliver	21
choke.time.f	choke point delay	exponential.f	choke.pt.wt	23
dispatcher	pt count error	uniform.f	-1, 1	7
"	to hosp time	get.travel.time	node dependent	13
"	to base time	"	"	17
find.hosp	hosp proportion	random.f	n/a	12
generator	inter-acc interval	nsp.f	n/a	1
"	prop trauma	random.f	n/a	22
get.amb	to acc time	get.travel.time	node dependent	14
get.loc	location	random.f	n/a	8
get.travel.time	travel time	mygamma.f	trav time, 3	per caller
get.patient	iv.rate	log.normal.f	120 - sbp,	
			10% cof.var	27
init.accident	no. victims	poisson.f	1.5	9
"	blunt v pen	random.f	n/a	28
init.pt	severe brain inj	random.f	n/a	25
"	hem v resp inj	random.f	n/a	26
"	iss	mybeta	1.390, 9.632	10
patient	resus time	mygamma.f	t.resus, 3	11
"	transfer time	exponential.f	t.tx	24
pvt.travel	hosp proportion	random.f	n/a	16
"	travel time	mygamma.f	1.2*trav time,	
			3	20

Table 1. Random variables used in the model.

Chapter 5

Verification and Validation

5.1 Verification

Major components of the implementation were verified against predictable model elements wherever possible. This was done by independent testing of "stub" routines where practical, and by inspection of the simulation trace or outputs elsewhere.

5.1.1 Random variate generators. Two new random number generators were implemented and verified; the non-stationary Poisson distribution routine `nsp.f`, and `mygamma.f`, a replacement for SIMSCRIPT's error-prone gamma variate generator.

5.1.1.1 `Nsp.f`. Çinlar's method [Çinlar75] of generating the interarrival times for a non-stationary Poisson arrival process was implemented in the function `nsp.f` (lines 2545 - 2578). An example test data set is given in Table 2, approximating the mean cumulative "arrivals" for a week. The `nsp.f` routine used this data to produce the 5000 arrival times summarized in Figure 3, Figure 4. The variation in generated arrival times closely follows the data in Table 2; a goodness of fit test shows no evidence of bad fit ($\chi^2_{13} = 13.752$, $P = .392$).

```

15      ;number of entries
0 0      ;first number is time,
12 10    ;second is number of events
24 5     ;in that time period
36 10    ;assumed to "wrap around" after reaching the last
48 5     ;time period
60 10
72 5
84 10
96 5
108 10
120 5
132 15
144 10
156 5
168 20

```

Table 2. Example data file of mean arrivals by interval for testing the nsp.f routine.

This process was repeated for a variety of data sets to establish acceptance of the nsp.f function.

5.1.1.2 Mygamma.f. A new gamma variate generator (lines 2442 - 2512) was implemented from two published algorithms [Bratley87]. For shape parameter greater than one, Tadikamalla's method was used, and for order one or less, Ahrens' method was used. Verification examples were produced over a wide range of arguments including those known to return invalid results for the SIMSCRIPT generator. An example of results for 2500 variates from mygamma.f given arguments 1.5, 1.5 ($\alpha = 1.5$, $\beta = 1.0$). This distribution has theoretical mean and variance equal to 1.5. The sample mean and variance of the output from mygamma.f was 1.517 and 1.529. Figure 5 presents a histogram of the output; the

Figure 4. Proportion arrivals in 12 hour periods over one week, corresponding to data in Table 2.

probability plots Figure 6 demonstrate the closeness to the theoretical distributional shape.

5.1.2 Static and Dynamic Analysis. Attributes of entities in the implementation were checked to confirm that they indeed matched the input parameters and distributional form specified in the model. For example, the distribution of observed ISS scores in the model compared reasonably well to that described by Baker [Baker92], which it was designed to match (Table 3). Similarly, the proportion of blunt to penetrating injury, the spatial distribution of injuries, the number of victims per accident, and other elements were confirmed to approximately match their inputs.

Figure 5. Histogram of 2500 variates from mygamma.f given arguments 1.5, 1.5.

statistic	model	Baker
N	1082	8791
mean	9.86	9.46
median	8.00	10.00
std deviation	7.30	7.18
skewness	1.26	1.71

Table 3. Characteristics of ISS scores obtained by the model and those reported by Baker.

The dynamic behavior of the implementation were verified to be compatible with the model by careful inspection of the

Figure 6. Probability plot of 2500 variates from mygamma.f given arguments 1.5, 1.5.

trace output and temporal outputs such as blood pressure. Special attention was paid to dispatching rules, such as alternating assignments between helicopter ambulances, or between two ambulances based in the same node. It was possible to confirm from the trace dispatched ambulances that were recalled had indeed not reached the scene. It was also confirmed that ambulances treated patients in order of

trace (edited for clarity)	comments
event 1 (acc) occurs in WJX at 0:42 acc 1 has pt 1 acc 1 has pt 2 acc 1 has pt 3 acc 1 has pt 4 D7 enroute to acc 1 at 0:44 D4 enroute to acc 1 at 0:44 D9 enroute to acc 1 at 0:44 new info fr event 1 at 0:46 D9 recalled at 0:46	Ambulances are dispatched in order of travel time to WJX. The last ambulance that has not yet arrived (D9) is recalled when more information on numbers of patients is available.
acc 25 has pt 26 acc 25 has pt 27 D28 treating pt 27, cts = 7.6 at 13:37 D42 treating pt 26, cts = 7.8 at 13:42 D28 enroute to UMC at 13:45 with 1 pts D42 enroute to UMC at 13:55 with 1 pts	Ambulances begin treating sicker patients (lower RTS) first. Both patients meet Level 1 criteria, so they are transported to UMC, not the nearest hospital.
STL went red at 12:11 D30 on scene at acc 139 in SSD at 12:14 D30 treating pt 165, cts = 12 at 12:16 D30 diverted from STL at 12:16 D42 on scene at acc 141 in JTB at 12:30 D42 treating pt 167, cts = 12 at 12:31 D42 diverted from STL at 12:31 D13 treating pt 168, cts = 12 at 12:31 D13 diverted from STL at 12:31 D20 diverted from STL at 12:32 D20 enroute to UMC at 12:32 with 1 pts D30 enroute to MMC at 12:37 with 1 pts D42 diverted from STL at 12:38 D42 enroute to UMC at 12:38 with 1 pts D13 diverted from STL at 12:40 D13 enroute to MMC at 12:40 with 1 pts STL went green at 13:11	St. Luke's goes on divert at 12:11. All subsequent runs which might routinely go to St. Luke's are diverted to alternate destinations until St. Luke's can begin accepting incoming ambulances again at 13:11

Table 4. Portions of trace output demonstrating the manifestation of specific model design items in the implementation.

severity as manifested by the current value of the RTS. And finally, the trace confirmed that no ambulance was dispatched to the "wrong" node or to the "wrong" hospital, and that no ambulance traveled to a hospital without carrying a patient. This method of verification can never absolutely confirm the reliability of the system, but it does serve to increase confidence that the implementation behaves according to the model's specifications. Table 4

shows a portion of the trace output that demonstrates the appearance in the implementation of several specific model behaviors.

5.2 Validation

Rigorous validation of a system such as this is extremely difficult, primarily because of the inadequacy of existing data sets useful for confirming model performance [McCoy92]. However, it was possible to compare measures of the model's performance to locally available data elements, to establish at least order of magnitude validity. The following items had sufficient data available to allow such comparisons: number of ambulance runs, number of helicopter runs, proportion of deaths prior to definitive care, etc. The model's predictions for these variables are compared with

item	model	Fire/Rescue	other
mean daily ground runs	41.734	44	n/a
mean daily helo runs	5.07	5	n/a
prob dead on scene	.018	.01	n/a
mean transport time (min, Duval Co only)	22.1	20	n/a
prob death prior to definitive care	.128	.05	.085 (Baker)
mean SBP at definitive care	93.8	100	95.4 (Baxt)

Table 5. Comparison of outcome estimates produced by the model with those estimated by Jacksonville Fire/Rescue.

convenience sample estimates from Jacksonville Fire Rescue and published data in Table 5. The distribution of transit times was compared with that derived from Campbell's data. The mean transit times were different, reflecting differing

geography, but quantile plots of the two data sets revealed that they have approximately the same differing only by a scaling factor (see Figure 7).

Figure 7. Quantile plot comparing the model's transport times with those provided by Campbell.

Chapter 6

Demonstrative Experiments

6.1 Triage Policy

To demonstrate the utility of the model in assessing policy choices, three sets of runs were performed using a different cutoff point to determine when a patient should be triaged directly to a Level 1 trauma center, bypassing other (possibly closer) hospitals. Current standard operating procedure calls for all patients with an RTS less than or equal to 90% of the maximum of 7.8408 (this corresponds to a score of approximately 10-11 on the 0-12 RTS scale) to be transported directly to a Level 1 center; this baseline case and several alternative cases were simulated. The minimum run length was set to 24 hours, and a total of 28 runs (approximately one month) were performed. The following outputs were used as measures of system performance under each scenario: trauma center utilization, red time, and reserve; proportion of accidents pended (i.e., no ambulance immediately available), mean waiting time until ambulance is dispatched among pended accidents; helicopter utilization; mean probability of death prior to receiving definitive care; unmet need (patients who met helicopter dispatch criteria but for whom a helicopter was unavailable); and total waiting time until EMS arrival (t_a). The classical

approach [Law91] was used to calculate point and interval estimates from the results of the 28 regeneration cycles. For this experiment, and the following one, output from the formatted data files was more useful than that directly reported by the program. An example of the direct output is provided in Appendix 4.

RTS cutoff (% of max)	80	90	95
trauma center utilization	.284±.035	.303±.031	.302±.034
red	.0	.0	.0016±.9E-5
reserve	.9998±.0003	.99997±.7E-5	.9997±.0006
pr acc pended	.041±.017	.054±.020	.040±.015
wait (min)	28.5±11.5	22.0±8.29	21.1±1.53
helicopter utilization	.111±.015	.136±.020	.149±.022
unmet need	.359±.056	.319±.045	.319±.045
pr death*	.120±.021	.128±.024	.121±.018
t _a	21.4±1.32	21.7±1.35	21.1±1.53
*prior to definitive care			

Table 6. System performance (mean ± 95% confidence interval) under different trauma center triage criteria.

The results for the baseline case and two alternatives (triage cutoffs of 80% and 95% of maximum) are summarized in Table 6. Compared to the baseline case, the main effects of liberalizing the triage cutoff are an increase in helicopter utilization, and a decrease in the length of time that a pended accident must wait to have an ambulance assigned to it.

triage cutoff convergence time (days)	helo utilization			waiting time		
	80	90	95	80	90	95
13.919	.128	.139	.163	34.2	21.0	16.3
30.928	.100	.133	.136	28.0	28.3	21.2

Table 7. Convergence points including at least a full seven day cycle.

The three alternative models converged at six points in the simulation; results at the two convergence points spanning at least a full week cycle are shown in Table 7. After adjusting for multiple comparisons, the results show that helicopter utilization is significantly different under the 80% and 95% triage cutoffs ($P = .013$, paired t test); the 95% confidence interval on the difference in utilization between these two alternatives is $.036 \pm .0014$, or about a 33% increase. Although the mean difference in waiting time for pended accidents is large between the 80% and 95% policies (12.3 minutes), the standard deviation of the difference is also large (7.81 minutes), so the results are not statistically significant. This result could be due to inadequate power since only two point estimates were obtained; further runs would be required to improve the precision of the estimate to determine if a true effect on waiting time should be expected.

6.2 Helicopter Dispatch Policy

Currently, helicopter ambulances are dispatched for patients needing a level 1 center whose transport time is over 19 minutes, and patients needing a level 2 center whose transport time is over 39 minutes. The effects of reducing these times by about 50% (to 10 and 20 minutes, respectively) are shown in Table 8; the triage cutoff was kept at 90% of maximum, so these results should be compared to the center column in Table 6. It appears that the effect of liberalizing time and distance transport criteria on helicopter utilization is much greater than that of liberalizing the triage cutpoint, yet the latter has received considerably more attention.

trauma center utilization	.298 ± .037
red	.000
reserve	.9999 ± .0001
pr acc pended	.042 ± .019
wait (min)	24.4 ± 8.86
helicopter utilization	.160 ± .021
unmet need	.293 ± .037
pr death*	.112 ± .020
time til arrival	22.2 ± 1.74
*prior to definitive care	

Table 8. System performance (mean ± 95% confidence interval) under alternate helicopter dispatch criteria.

6.3 Conclusion

It is interesting to note that the trauma triage cutoff, which has been the subject of vehement debate at times, had little effect on the overall load on the system, while a factor that has received little attention, the retriaging of less severely injured patients to a higher level of care if such a center is reasonably "close" had a much greater impact. This leads to the conclusion that the common knowledge of domain experts may not always be helpful in predicting the response of a complex system to change, and that computer models of such systems may enhance the decision makers accuracy and reliability by adding insight into the possible responses of the system to variables that were not previously thought important.

6.4 Further Work

Concern for the validity of current disaster planning and a demonstration of the potential of this model has led to community-wide interest in using a more fully validated version of the model to assist in planning for several events of importance in northeast Florida. The particular areas of interest are:

- a. Loss of a hospital and subsequent evacuation of its patients to other facilities.
- b. Loss of a major "choke point" such as a bridge for a period of hours to days.

- c. Widespread flooding of low areas eliminating multiple transportation routes and isolating some hospitals and nodes.
- d. An area-wide disaster such as a hurricane, which might combine all of the preceding elements.
- e. Modification of the physiologic model to use the a more detailed physiologic score such as ASCOT [Champion90], and to estimate the covariance structure of injuries from the American College of Surgeons National Trauma Registry Data (TRACS).

References

- [Baker74]
Baker, Susan P., O'Neill Brian, Haddon Jr., William, and Long, William. "The injury severity score: a method for describing patients with multiple injuries and evaluating emergency care." Journal of Trauma, 26, 3 (March 1974), pp. 187-196.
- [Baker92]
Baker, Susan P., O'Neill Brian, Ginsburg Marvin J., Li Guohua. The Injury Fact Book. Oxford University Press, Oxford, 1992.
- [Baxt87]
Baxt, William G., and Moody, Peggy. "The differential survival of trauma patients." Journal of Trauma, 27, 6 (June 1987), pp. 602-606.
- [Bratley87]
Bratley, B, Fox, BL, and Schrage, LE. A Guide to Simulation. Springer-Verlag, New York, 1987.
- [CACI88]
SIMGRAPHICS: User's Guide and Casebook. CACI Products Corporation, La Jolla, 1988.
- [Campbell92]
Campbell JP. Time-to-patient interval: the hidden component of response time [abstract]. Annals of Emergency Medicine 1992; 21:643.
- [Carter74]
Carter, Grace M. Simulation Model of Fire Department Operations: Program Description. The New York City Rand Institute, New York, 1974, pp. 8, 110-111.
- [Champion81]
Champion, Howard R., Sacco, William J., Carnazzo, Anthony J., Copes, Wayne, and Fouty, William J. "Trauma score." Journal of Trauma 9, 9 (September 1984), pp. 672-676.

- [Champion86]
 Champion, Howard R., Frey C. F., Sacco, W. J., et al.
 "Major trauma outcome study in quality assurance."
 Presented at the 4th Annual Meeting, Committee on
 Trauma, American College of Surgeons, Ft Lauderdale,
 FL, 1986.
- [Champion90]
 Champion, Howard R., Copes, W. S., Sacco, W. J., et al.
 A new characterization of injury severity. Journal of
 Trauma 1990; 30:539.
- [Chanpion91]
 Champion, Howard R., Sacco William J., Copes, Wayne S.
 "Trauma Scoring," in Moore, Ernest E., Mattox Kenneth
 L., and Feliciano David V. [editors], Trauma. Appleton
 & Lange, Norwalk, Connecticut, 1991.
- [Çinlar75]
 Cinlar, E. Introduction to Stochastic Processes.
 Prentice-Hall, Englewood Cliffs, NJ, 1975.
- [Devroye86]
 Devroye, Luc. Non-Uniform Random Variate Generation.
 Springer-Verlag, New York, 1986.
- [Fitzsimmons82]
 Fitzsimmons, James A., and Srikar, Bellur N.
 "Emergency ambulance location using the contiguous zone
 search routine." J Operations Management 2, 4 (August
 1982), pp. 225-237.
- [Johnson87]
 Johnson, Mark E. Multivariate Statistical Simulation.
 John Wiley & Sons, New York, 1987.
- [Law91]
 Law, Averill M., and Kelton, W. David, Simulation
 Modeling and Analysis. McGraw-Hill, New York, 1991.
- [Lewis79]
 Lewis, P. A. W., and Shedler, G. S. "Simulation of
 nonhomogeneous Poisson process by thinning." Nav Res
 Logist Quart, 26, (? 1979) pp.403-413.
- [Lewis86]
 Lewis, Frank R. Jr. "Prehospital intravenous fluid
 therapy: physiologic computer modelling." Journal of
 Trauma 26, 9 (September 1986), pp. 804-811.

[MacKenzie86]

MacKenzie, Ellen J., Shapiro, Sam, Moody, Mark, Siegel, John H., and Smith, Richard T. "Predicting posttrauma functional disability for individuals without severe brain injury." Medical Care, 24, 5 (May 1986), pp. 377-387.

[May90]

Adolf May. Traffic Flow Fundamentals. Englewood Cliffs, NJ, 1990.

[Mazzoni88]

Mazzoni, M. C., Borgström, P., Arfors, K. E., Intaglietta, M. "Dynamic fluid redistribution in hyperosmotic resuscitation of hypovolemic hemorrhage." American Journal of Physiology 255, 3 (Part 2, September 1988), pp. H629-H637.

[McCoy92]

McCoy, Lois Clark, S. Ruby, G. W. Reynolds, et al. "The hidden disaster in emergency management: The missing data base -- so you think there are data bases in notebooks waiting to be input into a computer." Abstract presented at 1992 Simulation MultiConference, Orlando, FL, 1992.

[Morris86]

Morris, John A., Auerbach, Paul S., Marshall, Gregory A., Bluth, Raymond F., Johnson, Lynda G., and Trunkey, Donald D. "The trauma score as a triage tool in the prehospital setting." Journal of the American Medical Association 256, 10 (September 12, 1986), pp. 1319-1325.

[Uyeno84]

Uyeno, Dean H., and Seeburg, C. "A practical methodology for ambulance location." Simulation 48, 2 (August 1984), pp. 79-87.

[Valenzuela90]

Valenzuela, Terrence D., Goldberg, Jeffery, Keeley, Kevin T., and Criss, Elizabeth A. "Computer modeling of emergency medical system performance." Annals of Emergency Medicine 19, 8 (August 1990), pp. 898-901.

[Wears90]

Wears, Robert L., and Winton, Charles N. "Load and go vs. stay and play: analysis of prehospital intravenous fluid therapy by computer simulation." Annals of Emergency Medicine 19, 2 (February 1990), pp. 163-168.

[Wilkinson90]

Wilkinson, Leland. SYSTAT: The System for Statistics.
SYSTAT, Inc., Evanston, IL, 1990.

[Zachariah92]

Pepe PE, Curka PA, Zachariah BS, et al. Urban trauma:
diurnal variations in the incidence, severity and
geographical distribution of various mechanisms of
injury [abstract]. Annals of Emergency Medicine 1992;
21:618.

Appendix 1

Program Source Code

Table of Contents

module	line number
preamble	2
main	418
accident	502
adj.time.f	541
ambulance.run	562
assign.amb	732
best.route	757
bleed	836
build.call.list	889
build.hosp.list	980
build.route	1077
check.accident	1100
check.red	1133
choke.f	1166
clear.reds	1184
cts.f	1208
dispatcher	1238
done	1395
final.report	1429
find.hosp	1482
find.loc	1538
ftime.f	1575
generator	1592
get.accs	1629
get.amb	1661
get.ems	1744
get.hosp	1783
get.iv.rate	1815
get.level	1826
get.list	1854
get.net	1877
get.num.amb	1933
get.patient	1945
get.sim	1990
get.table	2088
get.travel.time	2116
go.green	2142
go.off.red	2160
go.red	2186
init.accident	2205
init.pt	2262
initialize	2334
lin.int.f	2389
living	2414
mybeta.f	2427
mygamma.f	2443
no.pts	2516
nsp.f	2546
pass.time	2582

patient	2647
print.net	2760
pt.report	2869
pvt.travel	2893
read.call.list	2934
read.hosp.list	2955
recall	2977
resp.support	3014
run.report	3027
tr.check.in.acc	3108
tr.check.in.amb	3126
tr.check.in.pt	3138
tr.check.out.acc	3149
tr.check.out.pt	3167
tr.deliver.pt	3179
tr.enroute.hosp	3192
tr.go.green	3212
tr.go.red	3223
tr.on.scene	3234
tr.pickup.pt	3254
tr.pvt.tr.pt	3274
tr.resus.pt	3286
tr.send.amb	3300
tr.stack.acc	3320
tr.to.home	3338
tr.unstack.acc	3350
travel	3368
update	3381
write.call.list	3405
write.hosp.list	3426
	3443

Source Code

```

1 preamble      '' <t>      Prehospital Trauma Care Model
2 '' <s> preamble
3
4     normally, mode is undefined
5
6
7     '' functions and globals
8
9     define nsp.f, lin.int.f, adj.time.f, ftime.f, get.iv.rate
10    as double functions
11    define laplace.f, mygamma.f, mybeta.f, choke.f as double functions
12    define get.table, get.accs as pointer functions
13
14    define done, get.num.amb, get.level, find.loc, find.hosp, get.amb,
15    go.red, go.green, living as integer functions
16
17    define best.route, get.travel.time as double functions
18
19    define cum.events as a double, 2-dim array
20    define prop.accs as a double, 2-dim array
21
22    define min.length, pts.per.acc, alarm.lag, info.lag, start.time,
23    t.to.pt, t.on.scene, m.secure, s.secure, p.secure, t.resus, t.tx,
24    m.deliver, s.deliver, tr.prop, tmp.dur, tmp.ams, tmp.mn.pend
25    as double variables
26    define no.runs, run, num.helo, tmp.pts, tmp.accs, tmp.mx.pend,
27    no.deaths, no.blunt, no.pended as integer variables
28    define patient.counter, t.patient.counter, m.patient.counter,
29    tot.counter, acc.counter, med.counter, divert.counter,
30    override.counter, run.counter, red.limit, green.limit,
31    max.red.hosp, min.red, max.red, reds.per.day as integer variables
32    define TCRUISE, atol, htol, airspeed, range, clear.time, est.deaths,
33    r.est.deaths, ll.time, major.cutoff, minor.cutoff
34    as double variables
35    define min.amb, minor.time, major.time as integer variables
36    define nsp.tprime, nsp.last.time as double variables
37
38
39    '' entities and sets
40
41    permanent entities
42
43    every node has
44    a node.id,
45    a node.name,
46    a lat,
47    a long, and
48    owns an edge.set, a call.list, a hosp.list, a node.amb.set, and
49    may own a hosp.set, and
50    may belong to a node.set, an amb.base.set, a hosp.base.set,
51    a temp.set and an assignment.set
52    define node.id as an integer variable
53    define node.name as a text variable
54    define lat, long as double variables
55
56    '' could modify this to node, node, time.period to allow for
57    '' known changes in travel times by time of day
58    every node, node has
59    a transit.time,
60    a flight.time, and
61    owns a route

```

```

62      define transit.time, flight.time as double variables
63      define route as a lifo set
64
65      every hospital has
66          a hosp.id,
67          a hosp.name,
68          a no.pts,
69          a capacity,
70          a full,
71          a red,
72          a can.divert,
73          a red.today,
74          a red.start,
75          a clear.notice,
76          an r.max.pts,
77          an s.max.pts,
78          a gr.max.pts,
79          an update.time,
80          an r.accum.pts,
81          a g.accum.pts,
82          a g.ssq.pts,
83          an r.accum.cap,
84          a g.accum.cap,
85          a g.ssq.cap,
86          a hosp.base,
87          a hosp.volume,      'ann ED visits in 1000s
88          a level and
89      owns a resus.patient.set and
90      belongs to a hosp.set
91      may belong to the green.set and the red.set
92      define hosp.name as a text variable
93      define hosp.id, hosp.base, capacity, full, level, r.max.pts,
94          s.max.pts, gr.max.pts, red, can.divert, red.today
95          as integer variables
96      define hosp.volume, red.start, update.time, r.accum.pts,
97          g.accum.pts, r.accum.cap, g.accum.cap, r.ssq.pts, r.ssq.cap,
98          g.ssq.pts, g.ssq.cap as double variables
99      define no.pts as an integer variable monitored on the left
100     define clear.notice as a pointer variable
101
102     every ambulance has
103         a type,
104         an amb.id,
105         an amb.name,
106         an amb.run,
107         an amb.base,
108         a cur.location,
109         a travel.time and
110     may belong to the ready.set, the h.ready.set,
111     the out.of.service.set, the to.scene.set, the on.scene.set,
112     the to.hosp.set, the at.hosp.set, the to.base.set,
113     the at.base.set, the node.amb.set and the amb.set
114     define amb.name as a text variable
115     define type, amb.id, amb.base, cur.location as integer variables
116     define amb.run as a pointer variable
117     define travel.time as a double variable
118
119     temporary entities
120
121     every arc has
122         a source,
123         a sink,
124         a weight,

```

```

125         a choke.pt.wt,
126         a ch.cum.weight,
127         a cum.weight,
128         an arc.status, and
129 may belong to a route, a v.set, an mdt and an edge.set
130 define source, sink, arc.status as integer variables
131 define weight, choke.pt.wt, ch.cum.weight, cum.weight
132 as double variables
133
134 every call.item has
135     an ambulance.id,
136 and belongs to a call.list
137 define ambulance.id as an integer variable
138
139 every go.item has
140     a hospital.id,
141     a hosp.level
142 and belongs to a hosp.list
143 define hospital.id, hosp.level as integer variables
144
145
146 processes include generator
147
148     every ambulance.run has
149         a kind,
150         an ambulance.id,
151         a run.id,
152         a acc,
153         a src,
154         a destination,           ''destination is a node
155         a hosp,                 ''which is distinct fr a hospital
156         a dest.level,
157         a helo.coming,          ''flag
158         a status,
159         a scene.time and
160 owns an amb.patient.set
161 and may belong to an h.waiting.set
162 define kind, run.id, src, destination, dest.level, hosp, status,
163     helo.coming as integer variables
164 define acc as a pointer variable
165 define scene.time as a double variable
166
167 every patient has
168     an pt.id,
169     a condition,
170     a phase,
171     a change.flag,
172     a transp.mode,
173     an acc,
174     a hosp,
175     an injury.loc,
176     an injury.time,
177     a wait.time,
178     a scene.time,
179     an iv.start.time,
180     a transp.time,
181     a resus.time,
182     a trf.start.time,
183     a trf.rate,
184     a resp.time,
185     a tx.time,
186     a tod,
187     a pvt.tr.arrive,

```

```

188         a rescue.bp, a hosp.bp, a def.bp,
189         a r.cts,           '' Champion trauma score components
190         a h.cts,
191         a n.cts,
192         a cts.f function,
193         a cts,             '' revise tr score
194         an iss,           '' injury severity score
195         a prob.surv,
196         a bleeding.rate, a br.0,
197         a blood.volume, an sbp, an O2.sat,
198         an rbc.mass, a hct, an O2.delivery,
199         an iv.start.time, an iv.rate,
200         a blunt,          '' blunt v penetrating flag
201         a need.helo, a got.helo '' flags
202         and
203     belongs to the patient.set and
204     may belong to the pvt.tr.set, an acc.patient.set,
205         a resus.patient.set, and an amb.patient.set
206     define pt.id, condition, phase, transp.mode, r.cts, h.cts, n.cts,
207         iss, change.flag, blunt, injury.loc, need.helo,
208         got.helo as integer variables
209     define pvt.tr.arrive, injury.time, wait.time, resus.time, tx.time,
210         transp.time, br.0, iv.start.time, prob.surv, iv.rate,
211         trf.start.time, trf.rate, rescue.bp, hosp.bp, def.bp, tod,
212         resp.time, O2.sat, cts as double variables
213     define cts.f as a double function
214     define blood.volume, rbc.mass, sbp, bleeding.rate, hct, O2.delivery
215         as continuous double variables
216
217     '' note that accident is used for medical calls as well
218     every accident has
219         a kind,
220         an acc.id,
221         a site,
222         a no.victims,
223         a needed,
224         a sent,
225         an updated,           '' flag
226         a pended,            '' flag
227         an acc.start.time,
228         an acc.arrive.time and
229         an acc.end.time,
230         a blunt and
231     owns an acc.patient.set, an h.waiting.set and an amb.set, and
232     may belong to the pending.set and
233     may belong to the active.set
234     define acc.id, site, no.victims, needed, sent, updated, pended
235         as integer variables
236     define acc.start.time, acc.arrive.time, acc.end.time
237         as double variables
238     '' define acc.patient.set as a set ranked by low cts
239
240     '' events
241
242     event notices include clear.reds
243     every go.off.red has a gor.hsp
244     define gor.hsp as an integer variable
245     every resp.support has an r.pt
246     define r.pt as a pointer variable
247
248     the system owns
249         the amb.base.set, the hosp.base.set, the temp.set,
250         the assignment.set,

```

```

251         the node.set, the v.set, the mdt,
252         the patient.set, the pvt.tr.set,
253         the ready.set, the h.ready.set, the out.of.service.set,
254         the to.scene.set, the on.scene.set, the to.hosp.set,
255         the at.hosp.set, the to.base.set, the at.base.set,
256         the pending.set, the active.set,
257         the green.set and the red.set
258     define h.ready.set as a fifo set          ''helos alternate response
259
260     '' convenient defines
261
262     define TRUE to mean      1
263     define FALSE to mean    0
264     define NULL to mean     0
265     define FOREVER to mean until 1 = 2
266     define calls.pending to mean    n.pending.set > 0
267     define FREE to mean      0
268     define IN.USE to mean    1
269     define A.DUPLICATE to mean 2
270     define NMPD to mean      60          ''nautical miles / deg latitude
271
272     define DEBUG to mean     1 = 1      ''change to eliminate some output
273
274     '' process status for sta.a
275
276     define WRK to mean       1
277     define SSPND to mean     2
278     define NTRPT to mean     3
279
280     define ground to mean    1
281     define air to mean       2
282     define private to mean   3
283
284     define trauma to mean    1
285     define medical to mean   2
286
287     define alive to mean     0
288     define dead to mean      1
289     define sbp.0 to mean     120.0
290     define bv.0 to mean      4900.0
291     define hct.0 to mean     42.0
292     define O2.sat.0 to mean  1.0
293     define croak to mean     sbp.0 * hct.0 * O2.sat.0 / 4
294     define OPS to mean       1.6        ''Lewis constants
295     define PSTF to mean      .625
296     define TCONST to mean    25.0
297     define is.to.iv to mean   2.3      ''interstitial:intravascular ratio
298     define MAXRTS to mean    7.8408
299
300     '' messages
301     define req.amb to mean    1
302     define req.car to mean    2        ''private conveyance
303     define to.scene to mean   3
304     define at.scene to mean   4
305     define to.hosp to mean    5
306     define at.hosp to mean    6
307     define to.base to mean    7
308     define at.base to mean    8
309     define req.help to mean   10
310     define new.info to mean   11
311     define no.pts.left to mean 12
312     define req.helo to mean   13
313

```



```

314     '' ambulance run status
315     define working to mean 1
316     define not.working to mean 2
317
318     '' pt phase
319     define waiting to mean 1
320     define scene.rx to mean 2
321     define en.route to mean 3
322     define resus to mean 4
323     define pvt.trav to mean 5
324     define done.resus to mean 6
325
326     '' hospital capabilities
327     define level1 to mean 1
328     define level2 to mean 2
329     define level2a to mean 3 ''level 2 hosp that doesn't req tc stat
330     define level3 to mean 4
331
332
333     '' statistical counters
334
335     '' run duration
336
337     tally g.mean.dur as the mean,
338         g.var.dur as the variance, and
339         g.max.dur as the maximum of tmp.dur
340
341     '' pts and accidents
342
343     '' only trauma pts are tracked
344     tally r.no.t.pts as the runwise number of t.patient.counter
345     tally r.no.m.pts as the runwise number of m.patient.counter
346     tally r.no.accs as the runwise number of acc.counter
347     tally r.no.meds as the runwise number of med.counter
348     tally r.no.deaths as the runwise number of no.deaths
349     tally r.no.blunt as the runwise number of no.blunt
350
351     tally g.mean.pts as the mean,
352         g.var.pts as the variance, and
353         g.max.pts as the maximum of tmp.pts
354
355     tally g.mean.accs as the mean,
356         g.var.accs as the variance, and
357         g.max.accs as the maximum of tmp.accs
358
359     '' ambulance utilization and capacity
360
361     accumulate r.idle.amb as the runwise mean of n.ready.set
362     accumulate r.idle.helo as the runwise mean of n.h.ready.set
363
364     tally g.mean.amb as the mean, and
365         g.var.amb as the variance of tmp.ambs
366
367     '' queue for ambulance service
368
369     accumulate r.mean.pending as the runwise mean and
370         r.max.pending as the runwise maximum of n.pending.set
371
372     tally g.mean.pend as the mean, and
373         g.var.pend as the variance of tmp.mn.pend
374
375     tally g.mean.maxpend as the mean,
376         g.max.maxpend as the maximum, and

```

```

377         g.var.maxpend as the variance of tmp.mx.pend
378
379     '' hospitals
380     accumulate r.mean.red as the runwise mean of red
381
382     tally r.no.divert as the runwise number of divert.counter
383     tally r.no.override as the runwise number of override.counter
384     tally r.no.pended as the runwise number of no.pended
385
386     '' trace routines
387
388     '' ambulances
389     after filing in the ready.set,      call tr.check.in.amb
390     after filing in the h.ready.set,    call tr.check.in.amb
391     before filing in the to.scene.set,  call tr.send.amb
392     before filing in the on.scene.set,  call tr.on.scene
393     before filing in the to.hosp.set,   call tr.enroute.hosp
394     before filing in the at.hosp.set,   call tr.deliver.pt
395     before filing in the to.base.set,   call tr.to.home
396
397     '' accidents
398     before filing in the active.set,    call tr.check.in.acc
399     before filing in the pending.set,    call tr.stack.acc
400     after removing from the pending.set, call tr.unstack.acc
401     after removing from the active.set,  call tr.check.out.acc
402
403     '' patients
404     before filing in the acc.patient.set, call tr.check.in.pt
405     after removing from the patient.set,  call tr.check.out.pt
406     before filing in the amb.patient.set, call tr.pickup.pt
407     before filing in the pvt.tr.set,      call tr.pvt.tr.pt
408     before filing in the resus.patient.set, call tr.resus.pt
409
410     '' hospitals
411     after filing in the red.set,          call tr.go.red
412     after filing in the green.set,        call tr.go.green
413
414 end '' of preamble
415
416
417 main          '' <f>
418 '' <s> main
419
420     define mn as a double variable
421
422     open 4 for output, name is "summary.res"
423     use 4 for output
424     lines.v = 0
425
426     '' output data for statistical analysis
427     open 7 for output, name is "run.res"
428     use 7 for output
429     lines.v = 0
430
431     open 8 for output, name is "pt.res"
432     use 8 for output
433     lines.v = 0
434
435     use 6 for output
436     lines.v = 0
437
438     call initialize
439

```

```

440    '' outer loop to set up each arm of an experiment goes here
441
442    '' restore random number seeds
443
444    time.v = 0
445    nsp.tprime = 0
446    nsp.last.time = 0
447
448    for run = 1 to no.runs
449    do
450        reset runwise totals of t.patient.counter, m.patient.counter,
451            acc.counter, med.counter, divert.counter, override.counter,
452            n.ready.set, n.h.ready.set, n.pending.set, no.blunt,
453            no.deaths and no.pended
454        r.est.deaths = 0.
455        for each hospital
456        do
457            reset runwise totals of red(hospital)
458            r.accum.pts(hospital) = 0
459            r.accum.cap(hospital) = 0
460            r.max.pts(hospital) = 0
461        loop
462
463        start.time = time.v
464        schedule a clear.reds in clear.time hours
465        activate a generator now
466        start simulation
467
468        '' record runwise means here to ensure independence
469        tmp.dur = time.v - start.time
470        tmp.pts = r.no.t.pts
471        tmp.accs = r.no.accs
472        tmp.amb = 1.0 - r.idle.amb / n.ambulance    ''utilization
473        tmp.mn.pend = r.mean.pending
474        tmp.mx.pend = r.max.pending
475        for each hospital
476        do
477            mn = r.accum.pts(hospital) / tmp.dur
478            add mn to g.accum.pts(hospital)
479            add mn * mn to g.ssqs.pts(hospital)
480            mn = r.accum.cap(hospital) / tmp.dur
481            add mn to g.accum.cap(hospital)
482            add mn * mn to g.ssqs.cap(hospital)
483            add r.max.pts(hospital) to s.max.pts(hospital)
484            gr.max.pts(hospital) = max.f(gr.max.pts(hospital),
485                r.max.pts(hospital))
486        loop
487        call run.report
488
489        loop
490        call final.report    '' for this arm
491
492    '' end of outer loop
493
494    close unit 4
495    close unit 7
496    close unit 8
497
498 end '' of main
499
500
501 process accident          '' <f>18
502 '' <s> accident

```

```

503 '' create patients, assign their characteristics, and activate the ems system
504
505 define self, pt as pointer variables
506 define avg.cts, limit as double variables
507
508 self = accident
509
510 '' check pts to see if pvt tsp -- send them on
511 for each pt in acc.patient.set,
512     compute avg.cts as the avg of cts.f(pt)
513
514 '' empiric function for private transport
515 limit = .29 * ((avg.cts/7.8408)**6) + .01
516 wait 1 + exponential.f(alarm.lag - 1, 15) minutes
517
518 if random.f(2) >= limit or kind(self) = medical
519     call dispatcher giving NULL, req.amb, self
520
521     '' wait til accurate info is sent -- if ambulance arrives 1st,
522     '' it will force accurate info to be sent then
523
524     wait 1 + exponential.f(info.lag - 1, 3) minutes
525     call dispatcher giving NULL, new.info, self
526     updated(self) = TRUE
527
528     '' wait until all patients have been picked up
529     suspend
530 else '' go by pvt transport
531     call pvt.travel(self)
532 endif
533
534 remove self from the active.set
535 acc.end.time(self) = time.v
536
537 end '' of accident
538
539 function adj.time.f(t) '' <f>12
540 '' <s> adj.time.f
541 '' transit times for nodes assume cruising speed. This function adjusts
542 '' total travel time after Carter74 (taken from Kolesar and Walker 1974).
543 '' Their model assumes that a units speed gradually increases as it moves
544 '' progressively onto larger and larger roads, and then decreases as it
545 '' moves off thoroughfares to the accident scene, or the hospital.
546
547 define t as a double variable
548
549 if t <= 2 * TCRUISE '' never made it to cruising speed
550     t = 2 * sqrt.f(t * TCRUISE)
551 else
552     t = t + TCRUISE
553 endif
554
555 return with t
556
557 end '' of adj.time.f
558
559
560
561 process ambulance.run '' <f>
562 '' <s> ambulance.run
563
564 define this.ambulance as an integer variable
565 define self, this.accident, victim, other.run as pointer variables

```

```

566     define secure.time, null.time, cleanup.time, temp
567         as double variables
568
569     self = ambulance.run
570     this.ambulance = ambulance.id(self)
571     this.accident = acc(self)
572     status(self) = working
573     src(self) = cur.location(this.ambulance)
574     '' assign random variates here to preserve synchronization
575     if type(this.ambulance) = air
576         temp = t.on.scene / 2.0
577     else
578         temp = t.on.scene
579     endif
580     '' critical times adjusted to keep them above minimum step size in the
581     '' integrator routine -- shape parameter adjusted to make final returned
582     '' value approx a gamma(3)
583     scene.time(self) = 1 + mygamma.f(temp - 1, 3 * ((temp - 1) / temp)**2, 4)
584     null.time = exponential.f(t.to.pt, 5)
585     cleanup.time = exponential.f(3, 6)
586     secure.time = log.normal.f(m.secure, s.secure, 18)
587
588     '' travel to accident
589     call travel(cur.location(this.ambulance),
590         destination(self), this.ambulance)
591     if status(self) = working
592         ''check if recalled
593         call dispatcher giving this.ambulance, at.scene, this.accident
594
595         '' check if scene is secure
596         if kind(this.accident) = trauma
597             if random.f(19) < p.secure
598                 work secure.time minutes
599             endif
600         endif
601
602         '' work on scene
603         if acc.patient.set(this.accident) is not empty or
604             type(this.ambulance) = air or kind(this.accident) = medical
605             if type(this.ambulance) = air
606                 remove the first other.run from
607                 the h.waiting.set(this.accident)
608
609                 if sta.a(other.run) = WRK
610                     interrupt the ambulance.run called other.run
611                     add time.a(other.run) to scene.time(self)
612                     let time.a(other.run) = 0
613                 endif
614                 for each victim in amb.patient.set(other.run)
615                     do
616                         remove this victim from amb.patient.set(other.run)
617                         iv.rate(victim) = get.iv.rate(victim)
618                         transp.mode(victim) = type(this.ambulance)
619                         file this victim in amb.patient.set(self)
620                         dest.level(self) = min.f(dest.level(self),
621                             get.level(victim))
622                     loop
623
624                     dest.level(other.run) = level3
625                     helo.coming(other.run) = FALSE
626                     resume the ambulance.run called other.run
627                     work scene.time(self) minutes
628
629     else
630         ''ground

```

```

629
630     if kind(this.accident) = medical
631         dest.level(self) = level3      ''no categories for med pts
632         work.scene.time(self) minutes
633     else
634
635         work.null.time minutes
636
637         FOREVER
638         do
639             if acc.patient.set(this.accident) is not empty
640                 call get.patient(this.accident, self)
641
642                 if helo.coming = FALSE and
643                     cts(f.amb.patient.set(self)) > minor.cutoff *
644                         MAXRTS and
645                     acc.patient.set(this.accident) is not empty
646                         call get.patient(this.accident, self)
647                 endif
648
649                 if helo.coming(self) = FALSE
650                     work.scene.time(self) minutes
651                     leave
652                 endif
653
654                 '' compiler chokes id *endif-if* is changes to *els
655                 if helo.coming(self) = TRUE
656                     work.scene.time(self) minutes
657                     if helo.coming(self) = TRUE      '' if still wait
658                         suspend
659                     endif
660                     '' after helo arrival, its pt is gone so
661                     '' it loops around for another
662                 endif
663             endif
664             '' need second if statement because acc.patient.set
665             '' might have changed by now
666             if acc.patient.set(this.accident) is empty
667                 leave      '' no more pts
668             endif
669         loop
670     endif
671 endif
672
673 call check.accident(this.accident, self)
674
675 '' be sure you still have a pt, helo might have taken the last
676 if amb.patient.set(self) is not empty or kind(self) = medical
677     call dispatcher giving this.ambulance, to.hosp, this.accident
678
679     '' travel to hosp
680     if kind(self) = trauma
681         for every victim in amb.patient.set(self)
682         do
683             phase(victim) = en.route
684             hosp(victim) = hosp(self)
685             iv.rate(victim) = get.iv.rate(victim)
686             change.flag(victim) = TRUE
687         loop
688     endif
689     call travel(cur.location(this.ambulance),
690         destination(self), this.ambulance)
691

```

```

692         call dispatcher giving this.ambulance, at.hosp, this.accident
693
694         '' at hospital
695
696         work log.normal.f(m.deliver, s.deliver, 21) minutes
697         if kind(self) = trauma
698             for each victim in amb.patient.set(self)
699                 do
700
701                     phase(victim) = resus
702                     iv.rate(victim) = get.iv.rate(victim)
703                     remove this victim from amb.patient.set(self)
704                     file this victim in the resus.patient.set(hosp(victim))
705                     change.flag(victim) = TRUE
706                 loop
707             endif
708
709             '' cleanup
710             work cleanup.time minutes
711             remove this.ambulance from the at.hosp.set
712         endif
713     else
714         work null.time minutes      ''look around to be sure
715         call check.accident(this.accident, self)
716     endif
717 endif
718
719 '' after all that, return to base
720
721 call dispatcher giving this.ambulance, to.base, NULL
722 call travel(cur.location(this.ambulance),
723     destination(self), this.ambulance)
724
725 '' back at base
726 call dispatcher giving this.ambulance, at.base, NULL
727
728 end '' of ambulance.run
729
730
731 routine assign.amb(amb, accd)  '' <f>12
732 '' <s> assign.amb
733     define amb as an integer variable
734     define accd as a pointer variable
735
736     create an ambulance.run
737     kind(ambulance.run) = kind(accd)
738     ambulance.id(ambulance.run) = amb
739     amb.run(amb) = ambulance.run
740     acc(ambulance.run) = accd
741     destination(ambulance.run) = site(accd)
742     if type(amb) = air
743         dest.level(ambulance.run) = level2
744     else
745         dest.level(ambulance.run) = level3      ''any hospital
746     endif
747     add 1 to run.counter
748     file amb in amb.set(accd)
749     remove amb from the at.base.set
750     file amb in the to.scene.set
751     activate this ambulance.run now
752
753 end '' of assign.amb
754

```

```

755
756 function best.route given from.node, to.node      '' <f>18
757 '' <s> best.route
758 '' Uses Prim's algorithm to find shortest route from 'from' to 'to'.
759 '' uses choke pt weights as a penalty in deciding best route
760
761     define from.node, to.node as integer variables
762
763     define arc, best.arc, current.arc, check as pointer variables
764
765     '' drain working sets
766     for each arc in mdt          '' minimal distance tree
767         remove arc from mdt
768     for each node in node.set    '' nodes in tree so far
769         remove node from node.set
770     for each arc in v.set        '' candidate edges to add to tree
771         remove arc from v.set
772
773     '' initialize v.set
774     for each arc in edge.set(from.node)
775     do
776         file arc in v.set
777         cum.weight(arc) = weight(arc)
778         ch.cum.weight(arc) = weight(arc) + choke.pt.wt(arc)
779     loop
780
781     file from.node in node.set
782
783     '' process the graph to produce the minimal distance spanning tree
784     FOREVER
785     do
786         for each arc in v.set
787             compute best.arc as the min(arc) of ch.cum.weight(arc)
788
789         file best.arc in mdt
790
791         if sink(best.arc) = to.node
792             leave
793         endif
794
795         file sink(best.arc) in node.set
796         remove best.arc from v.set
797
798         for each arc in edge.set(sink(best.arc)),
799             when sink(arc) is not in node.set do
800             cum.weight(arc) = weight(arc) + cum.weight(best.arc)
801             ch.cum.weight(arc) = weight(arc) + choke.pt.wt(arc) +
802                 ch.cum.weight(best.arc)
803
804             for each check in v.set, '' check of new candidates for v.set
805                 with sink(check) = sink(arc)
806                 find the first case '' at most 1 case
807
808                 if found '' then sink(arc) is already a v.set destination
809                     if ch.cum.weight(arc) < ch.cum.weight(check)
810                         remove check from v.set '' arc is better than check
811                         file arc in v.set '' for this destination
812                     endif
813                 else '' arc gives a destination not already in v.set
814                     file arc in v.set
815                 endif
816             loop
817     loop

```



```

818
819     current.arc = best.arc
820     until source(current.arc) = from.node
821     do
822         call build.route(current.arc, from.node, to.node)
823         for each arc in mdt,
824             with sink(arc) = source(current.arc)
825                 find the first case
826         current.arc = arc
827     loop
828     call build.route(current.arc, from.node, to.node)
829
830     return with adj.time.f(cum.weight(best.arc))
831
832 end '' of best.route
833
834 routine bleed(patient)      '' <f>12
835 '' <s> bleed
836
837     define patient as a pointer variable
838     define x, n, d, rel.loss as double variables
839
840     if condition(patient) = dead      ''mark time
841         d.blood.volume(patient) = 0
842         d.sbp(patient) = 0
843         d.bleeding.rate(patient) = 0
844         d.rbc.mass(patient) = 0
845         d.hct(patient) = 0
846         d.O2.delivery(patient) = 0
847     else
848         x = 2**OPS
849
850         if iv.start.time(patient) > time.v
851             d.blood.volume(patient) = -bleeding.rate(patient) +
852                 iv.rate(patient) *
853                 (1 / (1 + is.to.iv) * (1.0 + is.to.iv *
854                     exp.f(-(time.v - iv.start.time(patient)) * hours.v * minutes.v /
855                         TCONST)))
856         else
857             d.blood.volume(patient) = -bleeding.rate(patient)
858         endif
859
860         '' derivative of the Lewis equation
861         rel.loss = 1 - blood.volume(patient) / bv.0
862         n = (x * abs.f(rel.loss)**0.6)
863         if rel.loss >= .5
864             list pt.id(patient), blood.volume(patient),
865                 d.blood.volume(patient), bleeding.rate(patient),
866                 d.bleeding.rate(patient), sbp(patient), d.sbp(patient)
867             trace
868         endif
869         d = (1 - sign.f(rel.loss) * x * abs.f(rel.loss)**1.6) ** .375
870         d.sbp(patient) = (sbp.0 / bv.0) * (n / d) * d.blood.volume(patient)
871
872         d.bleeding.rate(patient) = br.0 / sbp.0 * d.sbp(patient)
873         d.rbc.mass(patient) = -bleeding.rate(patient) * .01 * hct(patient)
874
875         d.hct(patient) = 100 * (blood.volume(patient) * d.rbc.mass(patient) -
876             rbc.mass(patient) * d.blood.volume(patient)) /
877             (blood.volume(patient)**2)
878
879         d.O2.delivery(patient) = O2.sat(patient) *
880

```

```

881          (sbp(patient) * d.hct(patient) + hct(patient) * d.sbp(patient))
882
883      endif
884
885  end '' of bleed
886
887
888  routine build.call.list      '' <f>12
889  '' <s> build.call.list
890  '' builds a default call list from the network structure based on
891  '' closest travel times -- each call list will contain at least min.amb
892  '' entries, and may contain more if they are within atol travel time of
893  '' the shortest travel time
894  '' assumes all helos go to all nodes
895
896  define i, there, here, count, ncount, amb as integer variables
897  define limit, tt as double variables
898
899  for each node called here
900  do
901      count = 0
902      for each node in the amb.base.set      ''copy the amb.base.set
903          file this node in the temp.set
904
905      FOREVER
906      do
907          for each node called i, with i in the temp.set
908          do
909              compute there as the minimum (i) of transit.time(here, i)
910          loop
911              remove there from the temp.set
912              file there in the assignment.set
913              tt = transit.time(here, there)
914              ''get all other nodes same distance away
915              for each node called i, with i in the temp.set and
916                  transit.time(here, i) = tt
917              do
918                  remove i from the temp.set
919                  file i in the assignment.set
920              loop
921
922              ncount = 0
923              if count = 0
924                  limit = (1 + atol) * tt
925              endif
926              for each node called i, with i in the assignment.set
927              do
928                  for each amb in the node.amb.set(i) with
929                      type(amb) = ground
930                  do
931                      add 1 to count
932                      add 1 to ncount
933                  loop
934              loop
935              if tt > limit
936                  if count - min.amb >= ncount
937                      leave
938                  endif
939              endif
940              for each node called i, with i in the assignment.set
941              do
942                  remove i from the assignment.set
943                  for each amb in node.amb.set(i) with type(amb) = ground

```

```

944         do
945             create a call.item
946             ambulance.id(call.item) = amb
947             '' be sure local ambulance always called first
948             if here = i
949                 file call.item first in the call.list(here)
950             else
951                 file call.item last in the call.list(here)
952             endif
953         loop
954     loop
955     if n.temp.set = 0
956         leave
957     endif
958     for each node in the assignment.set
959         remove this node from the assignment.set
960     loop
961     for each node in the temp.set
962         remove this node from the temp.set
963     for each node in the assignment.set
964         remove this node from the assignment.set
965
966     '' add helo to every call list (default)
967     for each amb in the h.ready.set
968         do
969             create a call.item
970             ambulance.id(call.item) = amb
971             file call.item in the call.list(here)
972         loop
973     loop
974 end '' of build.call.list
975
976 routine build.hosp.list    '' <f>12
977 '' <s> build.hosp.list
978 '' builds a default hospital list from the network structure based on
979 '' closest travel times -- each call list will contain at least one entry
980 '' for each level of care, and may contain multiple entries at the same level
981 '' if they are within atol travel time of the shortest travel time for that
982 '' level
983
984 define i, j, lvl, there, here, hosp, tot as integer variables
985 define count, first, finished as integer 1-dim arrays
986 define t as a double 1-dim array
987 define tt as a double variable
988 define NUMLEVELS to mean 3
989
990 reserve count, t, first, finished as NUMLEVELS
991
992 for each node called here
993 do
994     for i = 1 to NUMLEVELS
995     do
996         count(i) = 0
997         t(i) = 0.0
998         first(i) = TRUE
999         finished(i) = FALSE
1000     loop
1001     for each node in the hosp.base.set
1002         file this node in the temp.set
1003     for each node in the amb.base.set
1004         file this node in the temp.set
1005     for each node in the temp.set
1006         file this node in the temp.set

```

```

1007     FOREVER
1008     do
1009         for each node called j, with j in the temp.set
1010             do
1011                 compute there as the minimum (j) of transit.time(here, j)
1012             loop
1013             remove there from the temp.set
1014             if first(3) = TRUE
1015                 first(3) = FALSE
1016                 '' set max time to go to any level3
1017                 t(3) = transit.time(here, there) * (1 + htol)
1018                 finished(3) = TRUE ''dont care if we get any level3s
1019             endif
1020             for each hosp in the hosp.set(there)
1021                 do
1022                     tt = transit.time(here, there)
1023                     if level(hosp) >= level2a ''treat 2a like level2
1024                         lvl=level(hosp) - 1
1025                     else
1026                         lvl=level(hosp)
1027                     endif
1028                     if first(lvl) = TRUE
1029                         first(lvl) = FALSE
1030                         t(lvl) = tt * (1 + htol)
1031                         add 1 to count(lvl)
1032                         create a go.item
1033                         hospital.id(go.item) = hosp
1034                         hosp.level(go.item) = level(hosp)
1035                         if here = there
1036                             file go.item first in the hosp.list(here)
1037                         else
1038                             file go.item last in the hosp.list(here)
1039                         endif
1040                     else
1041                         if tt > t(lvl)
1042                             finished(lvl) = TRUE
1043                         else
1044                             add 1 to count(lvl)
1045                             create a go.item
1046                             hospital.id(go.item) = hosp
1047                             hosp.level(go.item) = level(hosp)
1048                             if here = there
1049                                 file go.item first in the hosp.list(here)
1050                             else
1051                                 file go.item last in the hosp.list(here)
1052                             endif
1053                         endif
1054                     endif
1055                 loop
1056             for i = 1 to NUMLEVELS
1057                 compute tot as the sum of finished(i)
1058             if tot = NUMLEVELS
1059                 leave '' found at least 1 hosp for level 1 & 2
1060             endif
1061             if n.temp.set = 0 '' no more candidate hospitals
1062                 leave
1063             endif
1064         loop
1065     for each node in the temp.set
1066         remove this node from the temp.set
1067     loop

```

```

1070
1071     release first, finished, t, count
1072
1073 end '' of build.hosp.list
1074
1075
1076 routine build.route given arc.id, from.node, to.node '' <f>18
1077 '' <s> build.route
1078 '' an arc can appear in at most 1 set named route
1079
1080     define arc.id, from.node, to.node as integer variables
1081
1082     if arc.status(arc.id) = FREE '' if arc.id is already in
1083         arc.status(arc.id) = IN.USE '' a route, a duplicate
1084         file arc.id in route(from.node, to.node) '' is created
1085     else
1086         create an arc
1087         source(arc) = source(arc.id)
1088         sink(arc) = sink(arc.id)
1089         weight(arc) = weight(arc.id)
1090         choke.pt.wt(arc) = choke.pt.wt(arc.id)
1091         arc.status(arc) = A.DUPLICATE '' marks arc as a duplicate
1092         file arc in route(from.node, to.node)
1093     endif
1094
1095     '' do we need to mark duplicates anymore?
1096 end '' of build.route
1097
1098
1099 routine check.accident(this.acc, this.run) '' <f>12
1100 '' <s> check.accident
1101 '' this function ensures that only the last ambulance to leave the
1102 '' scene destroys the accident process
1103
1104     define this.acc as a pointer variable
1105     define this.run as a pointer variable
1106
1107     define amb as an integer variable
1108
1109     amb = ambulance.id(this.run)
1110
1111     remove amb from the on.scene.set
1112     remove amb from the amb.set(this.acc)
1113
1114     if acc.patient.set(this.acc) is empty
1115         if h.waiting.set(this.acc) is empty
1116             acc(this.run) = NULL
1117             call dispatcher giving ambulance.id(this.run), no.pts.left,
1118                 this.acc)
1119         endif
1120     else '' more pts left
1121         if n.amb.set(this.acc) = 0
1122             '' if no one else is coming, call for help
1123             call dispatcher giving ambulance.id(this.run), req.help, this.acc
1124             '' this unit leaves now -- unmodelled engine co., police, etc
1125             '' remain on scene
1126         endif
1127     endif
1128
1129 end '' of check.accident
1130
1131
1132 routine check.red given hsp and n '' <f>15

```

```

1133 '' <s> check.red
1134 '' logic to place or remove hsp on divert status
1135
1136 define hsp, n as integer variables
1137 define cn as a pointer variable
1138
1139 if hsp is in the green.set
1140     if go.red(hsp, n) = TRUE
1141         remove hsp from the green.set
1142         file hsp in the red.set
1143         red(hsp) = TRUE
1144         add 1 to red.today(hsp)
1145         red.start(hsp) = time.v
1146         create a go.off.red called cn
1147         clear.notice(hsp) = cn
1148         schedule the go.off.red called cn giving hsp in min.red hours
1149     endif
1150 else
1151     if (time.v - red.start(hsp)) * hours.v > min.red
1152         if go.green(hsp, n) = TRUE
1153             cancel the go.off.red called clear.notice(hsp)
1154             clear.notice(hsp) = NULL
1155             remove hsp from the red.set
1156             file hsp in the green.set
1157             red(hsp) = FALSE
1158         endif
1159     endif
1160 endif
1161
1162 end '' of check.red
1163
1164
1165 function choke.f(here, there)          '' <f>12
1166 '' <s> choke.f
1167
1168 define here, there as integer variables
1169
1170 define result as a double variable
1171 define a as a pointer variable
1172
1173 result = 0.0
1174
1175 for each a in route(here, there), with choke.pt.wt(a) > 0.0
1176     add exponential.f(choke.pt.wt(a), 23) to result
1177
1178 return with result
1179
1180 end '' of choke.f
1181
1182
1183 event clear.reds          ''          <f>12
1184 '' <s> clear.reds
1185 '' clears all red status hospitals unconditionally
1186
1187 define i as an integer variable
1188
1189 for i = 1 to n.hospital
1190     red.today(i) = FALSE
1191 for each i in the red.set
1192 do
1193     remove i from the red.set
1194     file i in the green.set
1195     red(i) = FALSE

```

```

1196         if clear.notice(i) <> NULL
1197             cancel the go.off.red called clear.notice(i)
1198             clear.notice(i) = NULL
1199         endif
1200     loop
1201     schedule a clear.reds in 24.0 hours
1202     return
1203
1204 end '' of clear.reds
1205
1206
1207 function cts.f(pt)      '' <f>12
1208 '' <s> cts.f
1209
1210     define pt as a pointer variable
1211
1212     if sbp(pt) > 89
1213         h.cts(pt) = 4
1214     else
1215         if sbp(pt) > 75
1216             h.cts(pt) = 3
1217         else
1218             if sbp(pt) > 49
1219                 h.cts(pt) = 2
1220             else
1221                 if sbp(pt) > 30
1222                     h.cts(pt) = 1
1223                 else
1224                     h.cts(pt) = 0
1225                 endif
1226             endif
1227         endif
1228     endif
1229
1230     '' coeff from Champion HR, et al. J Trauma 89; 29:623
1231     cts(pt) = 0.2908 * r.cts(pt) + .7326 * h.cts(pt) + .9368 * n.cts(pt)
1232     return with cts(pt)
1233
1234 end '' of cts.f
1235
1236
1237 routine dispatcher given caller, message, this.accident      '' <f>
1238 '' <s> dispatcher
1239 '' serves as a monitor in this program
1240
1241     define res, caller and message as integer variables
1242     define this.accident, pt as pointer variables
1243
1244     define hsp, num as integer variables
1245
1246     select case message
1247     case req.amb      '' caller is not an ambulance
1248         if kind(this.accident) = medical
1249             needed(this.accident) = 1
1250         else
1251             num = int.f(no.victims(this.accident) + uniform.f(-1, 1, 7)
1252                 + .5)
1253             needed(this.accident) = get.num.amb(num)
1254         endif
1255
1256         '' find nearest ambulance(s)
1257         res = get.amb(this.accident, needed(this.accident), ground)
1258

```

```

1259 case req.help          '' caller is ambulance requesting more units
1260   if DEBUG
1261     write amb.name(caller), acc.id(this.accident),
1262           hour.f(time.v), minute.f(time.v) as
1263           " ", t 3, " req assist at acc ", i 5, " at ",
1264           i 2, ":", i 2, /
1265   endif
1266   needed(this.accident) =
1267     get.num.amb(n.acc.patient.set(this.accident))
1268   res = get.amb(this.accident, needed(this.accident), ground)
1269
1270 case req.helo
1271   if DEBUG
1272     write amb.name(caller), acc.id(this.accident),
1273           hour.f(time.v), minute.f(time.v) as
1274           " ", t 3, " req helo at acc ", i 5, " at ",
1275           i 2, ":", i 2, /
1276   endif
1277   res = get.amb(this.accident, 1, air)
1278   if res = air
1279     helo.coming(amb.run(caller)) = TRUE
1280     for each pt in amb.patient.set(amb.run(caller))
1281       got.helo(pt) = TRUE
1282     file amb.run(caller) in the h.waiting.set(this.accident)
1283     '' do not check to see if can recall a ground unit now
1284     '' 1st unit will be tied up til helo arrives, and addl units
1285     '' may be used to remove lower level pts
1286   else
1287     '' mark these pts as wanting a helo but cant get one
1288     for each pt in amb.patient.set(amb.run(caller))
1289       got.helo(pt) = FALSE
1290   endif
1291
1292 case new.info          '' 1st responder is updating w/ accurate info
1293   if DEBUG
1294     write acc.id(this.accident), hour.f(time.v), minute.f(time.v)
1295     as
1296     "new info fr event ", i 5, " at ", i 2, ":", i 2, /
1297   endif
1298   if kind(this.accident) = medical
1299     needed(this.accident) = 1
1300   else
1301     needed(this.accident) =
1302       get.num.amb(n.acc.patient.set(this.accident))
1303   endif
1304
1305   '' if too many ambulances have been sent
1306   if needed(this.accident) < sent(this.accident)
1307     call recall(sent(this.accident) - needed(this.accident),
1308               this.accident)
1309   endif
1310
1311   '' didn't send enough to begin with
1312   if needed(this.accident) > sent(this.accident)
1313     res = get.amb(this.accident,
1314                   needed(this.accident) - sent(this.accident), ground)
1315   endif
1316
1317   '' if just right, be sure acc isn't in pending set
1318   if needed(this.accident) = sent(this.accident) and
1319     this.accident is in the pending.set
1320     remove this.accident from the pending.set
1321   endif

```



```

1322
1323 case at.scene
1324     remove caller from the to.scene.set
1325     file caller in the on.scene.set
1326
1327     '' if this ambulance is the first to arrive
1328     if acc.arrive.time(this.accident) = 0
1329         acc.arrive.time(this.accident) = time.v
1330         '' if update hasn't been sent yet, force it to be
1331         '' sent now
1332         if updated(this.accident) = FALSE
1333             interrupt the accident called this.accident
1334             time.a(this.accident) = 0
1335             resume the accident called this.accident
1336         endif
1337     endif
1338
1339 case no.pts.left
1340     '' reactivate the accident, allowing it to end
1341     '' a DESTROY may not free allocated memory, per CACI
1342     if this.accident is in the pending.set
1343         remove this.accident from the pending.set
1344     endif
1345     if amb.set(this.accident) is not empty
1346         call recall(n.amb.set(this.accident), this.accident)
1347     endif
1348     '' cant use an else here, because the recall function might
1349     '' cause the amb.set to become empty
1350     if amb.set(this.accident) is empty
1351         resume the accident called this.accident
1352     endif
1353
1354 case to.hosp
1355     '' find receiving hosp
1356     hsp = find.hosp(caller)
1357     travel.time(caller) = get.travel.time(cur.location(caller),
1358         hosp.base(hsp), type(caller), 13)
1359     hosp(amb.run(caller)) = hsp
1360     file caller in the to.hosp.set
1361
1362 case at.hosp
1363     remove caller from the to.hosp.set
1364     file caller in the at.hosp.set
1365
1366 case to.base
1367     file caller in the to.base.set
1368     destination(amb.run(caller)) = amb.base(caller)
1369     travel.time(caller) = get.travel.time(cur.location(caller),
1370         amb.base(caller), type(caller), 17)
1371
1372 case at.base
1373     remove caller from the to.base.set
1374     file caller in the at.base.set
1375     if type(caller) = ground
1376         file caller in the ready.set
1377         if calls.pending
1378             remove the first accident from the pending.set
1379             res = get.amb(accident, 1, ground) '' should be only 1 in
1380         endif
1381     else
1382         file caller in the h.ready.set
1383     endif
1384

```

```

1385         default
1386             print 1 line with message thus
1387             Invalid message ***
1388
1389     endselect
1390
1391 end '' of dispatcher
1392
1393
1394 function done(patient)      '' <f>12
1395 '' <s> done
1396
1397     define patient as a pointer variable
1398
1399     if condition(patient) = alive      ''if just now died
1400         if living(patient) = FALSE
1401             return with TRUE
1402         endif
1403     endif
1404
1405     if phase(patient) = pvt.trav
1406         if time.v > pvt.tr.arrive
1407             return with TRUE
1408         endif
1409     endif
1410
1411     if phase(patient) = resus
1412         if time.v >= injury.time(patient) + (wait.time(patient) +
1413             transp.time(patient) + scene.time(patient) +
1414             resus.time(patient)) / hours.v / minutes.v
1415             return with TRUE
1416         endif
1417     endif
1418
1419     if change.flag(patient) = TRUE
1420         return with TRUE
1421     endif
1422
1423     return with FALSE
1424
1425 end '' of done
1426
1427
1428 routine final.report      '' <f>18
1429 '' <s> final.report
1430
1431     define correction as a double variable
1432
1433     use 4 for output
1434     correction = no.runs / (no.runs - 1)
1435
1436     print 8 lines with no.runs, min.length,
1437         g.mean.dur, sqrt.f(g.var.dur * correction), g.max.dur,
1438         g.mean.accs, sqrt.f(g.var.accs * correction), g.max.accs,
1439         100 * no.blunt / (no.runs * g.mean.accs),
1440         g.mean.pts, sqrt.f(g.var.pts * correction), g.max.pts,
1441         no.deaths / no.runs and est.deaths / no.runs
1442     thus
1443 Results after ** runs of at least **.** days per run
1444         average      sd (of runwise means)      max
1445 duration      ***.**      ***.**      ***.**
1446 no. accidents      ***.*      ***.**      ***
1447 % blunt      ***.*

```

```

1448 no. tr. patients      ***.*      ***.**      ***
1449 no. deaths           ***.*
1450 no. est deaths       ***.*
1451
1452 print 4 lines with g.mean.amb, sqrt.f(g.var.amb * correction),
1453 g.mean.pend, sqrt.f(g.var.pend * correction), g.mean.maxpend,
1454 g.max.maxpend thus
1455 amb. util             *.**      *.***
1456 amb. queue           *.***      *.****      mean ***.*
1457                                           glob ***
1458
1459 print 3 lines thus
1460 hospital utilization
1461 name      load      max      reserve
1462      avg      sd      avg      global      avg      sd
1463 for each hospital
1464 do
1465     print 1 line with hosp.name(hospital),
1466     g.accum.pts(hospital) / no.runs,
1467     sqrt.f((g.ss.pts(hospital) - (g.accum.pts(hospital) *
1468     g.accum.pts(hospital) / no.runs)) / (no.runs - 1)),
1469     s.max.pts(hospital) / no.runs, gr.max.pts(hospital),
1470     1 - g.accum.cap(hospital) / no.runs,
1471     sqrt.f((g.ss.cap(hospital) - (g.accum.cap(hospital) *
1472     g.accum.cap(hospital) / no.runs)) / (no.runs - 1))
1473     thus
1474     ***      **.*      **.*      ***      ***      *.***      *.****
1475 loop
1476 use 6 for output
1477
1478 end '' of final.report
1479
1480
1481 function find.hosp(caller)      '' <f>12
1482 '' <s> find.hosp
1483 '' chooses a destination hospital for the ambulance caller
1484
1485 define caller as an integer variable
1486
1487 define item as a pointer variable
1488 define limit as a double variable
1489 define hsp, got.it, cumtot, rtot, first.choice as integer variables
1490
1491 limit = random.f(12)
1492 got.it = FALSE
1493 first.choice = NULL
1494
1495 for each item in hosp.list(cur.location(caller)),
1496     with hosp.level(item) <= dest.level(amb.run(caller)) and
1497     hospital.id(item) in the green.set
1498     compute cumtot as the sum of hosp.volume(hospital.id(item))
1499
1500 for each item in hosp.list(cur.location(caller)),
1501     with hosp.level(item) <= dest.level(amb.run(caller))
1502 do
1503     if first.choice = NULL
1504         first.choice = hospital.id(item)
1505     endif
1506     if hospital.id(item) is in the red.set
1507         add 1 to divert.counter
1508         if DEBUG
1509             write amb.name(caller), hosp.name(hospital.id(item)),
1510             hour.f(time.v), minute.f(time.v) as

```

```

1511             t 3, " diverted from ", t 3, " at ", i 2, ":", i 2, /
1512         endif
1513         cycle      ''look for another
1514     endif
1515     hsp = hospital.id(item)
1516     add hsp.volume(hsp) to rtot
1517     if rtot / cumtot >= limit
1518         got.it = TRUE
1519         leave
1520     endif
1521 loop
1522
1523 if got.it = FALSE    ''appr hsp is red & no alternate, so override
1524     add 1 to override.counter
1525     hsp = first.choice
1526     if DEBUG
1527         write amb.name(caller), hour.f(time.v), minute.f(time.v) as
1528             t 3, " overrides divert at ", i 2, ":", i 2, /
1529     endif
1530 endif
1531
1532 return with hsp
1533
1534 end '' of find.hosp
1535
1536 function find.loc(amb)      '' <f>12
1537 '' <s> find.loc
1538
1539     define amb as an integer variable
1540
1541     define prop, dist, x, y as double variables
1542     define link as a pointer variable
1543     define ans as an integer variable
1544
1545     prop = 1.0 - time.a(amb.run(amb)) / travel.time(amb)
1546     dist = 0.0
1547     x = 0.0
1548     for every link in route(src(amb.run(amb)),
1549         destination(amb.run(amb)))
1550         add weight(link) to dist
1551     let dist = dist * prop      '' distance traveled so far
1552     for every link in route(src(amb.run(amb)),
1553         destination(amb.run(amb)))
1554     do
1555         y = weight(link) / 2.0
1556         add y to x
1557         if x > dist
1558             ans = src(amb.run(amb))
1559             leave
1560         endif
1561         add y to x
1562         if x > dist
1563             ans = destination(amb.run(amb))
1564             leave
1565         endif
1566     loop
1567
1568 return with ans
1569
1570 end '' of find.loc
1571
1572
1573

```

```

1574 function ftime.f(f, t)          '' <f>12
1575 ''<s> ftime.f
1576
1577     define f, t as integer variables
1578
1579     define temp1, temp2, av.lat as double variables
1580
1581     temp1 = (lat(f) - lat(t)) * NMPD
1582     av.lat = (lat(f) + lat(t)) / 2
1583     temp2 = (long(f) - long(t)) * cos.f(av.lat) * NMPD
1584
1585     return with ll.time +
1586         sqrt.f(temp1*temp1 + temp2*temp2) / airspeed * minutes.v)
1587
1588 end '' of ftime.f
1589
1590
1591 process generator          '' <f>15
1592 '' <s> generator
1593 '' system starts empty and idle, runs min.length days and stops when it
1594 '' is next empty and idle
1595
1596     '' note use of weekday.f instead of day.f -- if desire to model
1597     '' seasonality, etc, must call origin.r with a starting date, and
1598     '' then use day.f throughout
1599     write run, weekday.f(time.v), hour.f(time.v), minute.f(time.v) as
1600         "run ", i 5, " starts at day ", i 1, " at ", i 2, ":", i 2, /
1601
1602     if run = 1
1603         wait nsp.f(cum.events(*, *), 1) hours
1604     endif
1605
1606     FOREVER
1607     do
1608         if (time.v - start.time) > min.length and
1609             r.no.accs > 0 and
1610             the patient.set is empty and
1611             n.ambulance = n.ready.set + n.h.ready.set and
1612             the red.set is empty
1613             leave
1614         endif
1615         create an accident
1616         add 1 to tot.counter
1617         acc.id(accident) = tot.counter
1618         call init.accident(accident)
1619         activate accident now
1620         wait nsp.f(cum.events(*, *), 1) hours
1621     loop
1622     clear.time = (time.a(clear.reds) - time.v) * hours.v
1623     cancel clear.reds
1624
1625 end '' of generator
1626
1627
1628 function get.accs          '' <f>12
1629 '' <s> get.accs          creates table representing the proportion of
1630 ''                        accidents occurring at each node
1631
1632     define table as a double, 2-dim array
1633     define num, cum as double variables
1634     define i as an integer variable
1635
1636     reserve table(*, *) as n.node by 2

```

```

1637
1638 open 3 for input, name is "space.dat"
1639 use 3 for input
1640 cum = 0.0
1641
1642 for i = 1 to n.node
1643 do
1644     read num                '' number of events in that node
1645     add num to cum          '' per some unit of time
1646     table(i, 2) = cum      '' as a cumulative distribution
1647 loop
1648 close unit 3
1649
1650 for i = 1 to n.node        '' normalize the table
1651 do
1652     table(i, 2) = table(i, 2) / cum
1653 loop
1654
1655 return with table(*, *)
1656
1657 end '' of get.accs
1658
1659
1660 function get.amb given acc, num, and kno      '' <f>18
1661 '' <s> get.amb
1662 '' finds nearest ambulance(s)
1663 '' if can't dispatch no. of ground ambulances requested, the accident is
1664 '' placed on the pending list.
1665 '' returns type of ambulance sent for notification of caller
1666
1667 define acc as a pointer variable
1668 define num, kno, amb, amb1 as integer variables
1669
1670 define item, item1 as pointer variables
1671 define result, target, b, found.count as integer variables
1672
1673 target = site(acc)
1674
1675 for each item in call.list(target), with
1676     (ambulance.id(item) in the ready.set or
1677     ambulance.id(item) in the h.ready.set)
1678     and type(ambulance.id(item)) = kno
1679 do
1680     amb = ambulance.id(item)
1681     select case kno
1682     case ground
1683         ''handle the rare case of two ambs based at one node
1684         if n.node.amb.set > 1
1685             b = amb.base(amb)
1686             for each amb1 in the ready.set,
1687                 with amb.base(amb1) = b
1688                 and type(amb1) = kno          ''guaranteed to have 1
1689             do
1690                 ''first amb in the ready set that is in the call list
1691                 ''is the next up for a run
1692                 for each item1 in call.list(target), with
1693                     ambulance.id(item1) = amb1
1694                 find the first case
1695                 if found
1696                     leave
1697                 endif
1698             loop
1699             item = item1

```

```

1700         amb = amb1
1701     endif
1702     remove amb from the ready.set
1703
1704
1705     case air    'helos alternate call, but are available to all nodes
1706         if ambulance.id(item) = f.h.ready.set
1707             remove ambulance.id(item) from the h.ready.set
1708         else
1709             cycle
1710         endif
1711     endselect
1712     travel.time(amb) = get.travel.time(target, cur.location(amb),
1713         type(amb), 14)
1714     call assign.amb(amb, acc)
1715     add 1 to sent(acc)
1716     add 1 to found.count
1717     if found.count = num
1718         leave
1719     endif
1720 loop
1721
1722     if found.count < num
1723         result = NULL
1724     else
1725         result = knd
1726     endif
1727
1728     if knd = ground
1729         if result = NULL and acc is not in the pending.set
1730             file acc in the pending.set
1731             if pended(acc) = FALSE
1732                 pended(acc) = TRUE
1733                 add 1 to no.pended
1734             endif
1735         endif
1736     endif
1737
1738     return with result
1739
1740 end '' of get.amb
1741
1742
1743 routine get.ems    '' <f>12
1744 ''<s>    get.ems
1745 ''    ambulance capability ignored -- no explicit modeling of
1746 ''    first responders -- amb's are assumed to be ACLS-paramedic level
1747
1748     define i as an integer variable
1749
1750     open 3 for input, name is "amb.dat"
1751     use 3 for input
1752     read n.ambulance
1753     start new record
1754     create each ambulance
1755     i = 1
1756     for each ambulance
1757     do
1758         amb.id(ambulance) = i
1759         read type(ambulance), amb.base(ambulance), amb.name(ambulance)
1760         start new record
1761         cur.location(ambulance) = amb.base(ambulance)
1762         file this ambulance in the node.amb.set(amb.base(ambulance))

```

```

1763         if amb.base(ambulance) is not in the amb.base.set
1764             file amb.base(ambulance) in the amb.base.set
1765         endif
1766         if type(ambulance) = ground
1767             file this ambulance in the ready.set
1768         endif
1769         if type(ambulance) = air
1770             file this ambulance in the h.ready.set
1771             add 1 to num.helo
1772         endif
1773         file this ambulance in the at.base.set
1774         add 1 to i
1775     loop
1776
1777     close unit 3
1778
1779 end '' of get.ems
1780
1781
1782 routine get.hosp          '' <f>12
1783 '' <s> get.hosp
1784
1785     define i as an integer variable
1786
1787     open 3 for input, name is "hosp.dat"
1788     use 3 for input
1789     read n.hospital
1790     start new record
1791     create each hospital
1792     i = 1
1793     for each hospital
1794     do
1795         hosp.id(hospital) = i
1796         read hosp.name(hospital), hosp.base(hospital), hosp.volume(hospital),
1797             level(hospital), capacity(hospital), can.divert(hospital)
1798         start new record
1799         file this hospital in the green.set
1800         file this hospital in hosp.set(hosp.base(hospital))
1801         if hosp.base(hospital) is not in the hosp.base.set
1802             file hosp.base(hospital) in the hosp.base.set
1803         endif
1804         full(hospital) = FALSE
1805         red(hospital) = FALSE
1806         red.today(hospital) = FALSE
1807         add 1 to i
1808     loop
1809     close unit 3
1810
1811 end '' of get.hosp
1812
1813
1814 function get.iv.rate(pt)    '' <f>12
1815 '' <s> get.iv.rate
1816
1817     define pt as a pointer variable
1818
1819     return with (10 + log.normal.f(2.0 * max.f(.5, sbp.0 - sbp(pt)),
1820         .2 * max.f(.5, sbp.0 - sbp(pt)), 27)) * hours.v * minutes.v
1821
1822 end '' of get.iv.rate
1823
1824
1825 function get.level(pt)     '' <f>12

```



```

1826 '' <s> get.level
1827 '' logic to assign patient to a level of care based SOLELY on pt
1828 '' characteristics. dispatcher may choose to modify based on travel
1829 '' time, load, availability, etc.
1830
1831     define pt as a pointer variable
1832
1833     define lvl as an integer variable
1834
1835     '' this logic may (should) be modified later, and should be
1836     '' modifiable at runtime w/o recompilation
1837
1838     if cts(pt) <= major.cutoff
1839         lvl = level1
1840     else
1841         if cts(pt) > minor.cutoff
1842             lvl = level3
1843         else
1844             lvl = level2a
1845         endif
1846     endif
1847
1848     return with lvl
1849
1850 end '' of get.level
1851
1852
1853 routine get.list given fname, readprog, buildprog, writeprog    ''<f>12
1854 '' <s> get.list
1855 '' if no call list is provided from a file, builds a default call list for
1856 '' ground ambulance based on transit times, and writes results to a file
1857 '' for subsequent editing as needed
1858
1859     define fname as a text variable
1860     define readprog, buildprog, writeprog as subprogram variables
1861
1862     ''if call list file exists, read it in
1863     open unit 3 for input, name is fname, noerror
1864     use 3 for input
1865     if ropenerr.v eq FALSE
1866         call readprog
1867     else ''else build default call list of at least min.amb
1868         close unit 3
1869         call buildprog
1870         call writeprog
1871     endif
1872
1873 end '' of get.call.list
1874
1875
1876 routine get.net          '' <f>12
1877 '' <s> get.net
1878
1879     define from.node, to.node as integer variables
1880     define temp as a double variable
1881
1882     open 3 for input, name is "net.dat"    ''transportation network
1883     use 3 for input
1884     read n.node
1885     start new record
1886     create each node
1887
1888     for each node

```

```

1889     do
1890         read node.id(node), node.name(node), lat(node), long(node)
1891         start new record
1892     loop
1893
1894     read airspeed, range
1895     start new record
1896
1897     while data is not ended
1898     do
1899         create an arc
1900         read source(arc), sink(arc), weight(arc), choke.pt.wt(arc)
1901         file arc in the edge.set(source(arc))
1902     loop
1903
1904     close unit 3
1905
1906     '' calculate transit times
1907     for each node called from.node
1908     do
1909         for each node called to.node
1910         do
1911             if from.node = to.node          '' calc intranode transit time
1912                 temp = 0.0
1913                 for each arc in edge.set(from.node)
1914                 do
1915                     add weight(arc) to temp
1916                 loop
1917                 temp = 0.5 * temp / n.edge.set(from.node)
1918                 transit.time(from.node, from.node) = adj.time.f(temp)
1919                 flight.time(from.node, from.node) = ll.time +
1920                     (temp / airspeed * minutes.v)
1921             else
1922                 transit.time(from.node, to.node) =
1923                     best.route(from.node, to.node)
1924                 flight.time(from.node, to.node) = ftime.f(from.node, to.node)
1925             endif
1926         loop
1927     loop
1928
1929 end '' of get.net
1930
1931 function get.num.amb(n)    '' <f>12
1932 '' <s> get.num.amb
1933 '' logic to determine the no. of ambulances to be dispatched
1934 '' based on est no. of patients, their severity, number dispatched already
1935
1936     define n as an integer variable
1937
1938     return with (max.f(1, int.f(n / 2 + .5)))
1939
1940 end '' of get.num.amb
1941
1942
1943 routine get.patient(acc, amb.run)    '' <f>12
1944 '' <s> get.patient
1945
1946     define acc, amb.run as pointer variables
1947
1948     define v, victim as a pointer variable
1949     define temp, poss.dest as integer variables
1950     define t as a double variable
1951

```

```

1952
1953     for each v in the acc.patient.set(acc)
1954     do
1955         compute victim as the min(v) of cts.f(v)
1956         '' this should update every pts cts
1957     loop
1958     remove victim from acc.patient.set(acc)
1959     phase(victim) = scene.rx
1960     transp.mode(victim) = type(ambulance.id(amb.run))
1961     file victim in amb.patient.set(amb.run)
1962     temp = get.level(victim)
1963     dest.level(amb.run) = min.f(dest.level(amb.run), temp)
1964     iv.start.time(victim) = time.v + (scene.time(amb.run) /
1965         hours.v / minutes.v) / 2
1966     resp.time(victim) = scene.time(amb.run) / 3
1967     schedule a resp.support(victim) in resp.time(victim) minutes
1968     iv.rate(victim) = get.iv.rate(victim)
1969
1970     '' should helo be requested? if so, pt is still assigned to
1971     '' amb for scene rx, but flag is set to make him wait for helo arrival
1972     poss.dest = find.hosp(ambulance.id(amb.run))
1973     if temp < level3
1974         if temp = level1
1975             t = major.time
1976         else
1977             t = minor.time
1978         endif
1979         if transit.time(site(acc), hosp.base(poss.dest)) > t
1980             need.helo(victim) = TRUE
1981             call dispatcher giving ambulance.id(amb.run), req.helo, acc
1982         endif
1983     endif
1984     change.flag(victim) = TRUE
1985
1986 end '' of get.patient
1987
1988
1989 routine get.sim      '' <f>12
1990 '' <s> get.sim
1991
1992     define i, no.streams as integer variables
1993
1994     open 3 for input, name is "ems.dat"
1995     use 3 for input
1996
1997     read no.runs
1998     start new record
1999     read min.length      ''min run length in days
2000     start new record
2001     read tr.prop         ''trauma proportion
2002     start new record
2003     read pts.per.acc
2004     start new record
2005     read alarm.lag      ''initial alarm latency in minutes
2006     start new record
2007     read info.lag       ''delay til first responder reports
2008     start new record
2009     read m.secure, s.secure ''mean & sd secure time
2010     start new record
2011     read p.secure       ''prob of needing secure time
2012     start new record
2013     read t.to.pt        ''time til patient
2014     start new record

```

```

2015 read t.on.scene          ''scene treatment
2016 start new record
2017 read m.deliver, s.deliver ''meand & sd deliver time
2018 start new record
2019 read t.resus            ''mean resus time in minutes
2020 start new record
2021 read t.tx              ''tx to def care time
2022 start new record
2023 read TCRUISE           ''time in minutes til reach cruising speed
2024 start new record
2025 read min.amb           ''minimum # of amb's on a call list
2026 start new record
2027 read atol             ''proportional tolerance in dist to amb's
2028 start new record
2029 read htol             ''proportional tolerance in dist to hospitals
2030 start new record
2031
2032 read minor.cutoff       ''cts >= minor.cutoff will ride double
2033 minor.cutoff = MAXRTS * minor.cutoff
2034 start new record
2035 read major.cutoff       ''cts <= major goes to level 1
2036 major.cutoff = MAXRTS * major.cutoff
2037 start new record
2038 read minor.time        ''> minor.time, call helo
2039 start new record
2040 read major.time        ''> major.time, call helo
2041 start new record
2042 read ll.time           ''mean launch + land time for helo
2043 start new record
2044 ''no. over (under) cap to go red (green), and
2045 ''max no. hosps that can be red simultaneously
2046 read red.limit, green.limit, max.red.hosp
2047 start new record
2048 read min.red, max.red, reds.per.day ''min & max red.time, no. of reds/day
2049 start new record      ''allowed
2050 read clear.time       ''time of day for clearing red.status
2051
2052 close unit 3
2053
2054 open 3 for input, name is "cont.dat"    ''continous simulation control
2055 use 3 for input
2056 read max.step.v
2057 start new record
2058 read min.step.v
2059 start new record
2060 read abs.err.v
2061 start new record
2062 read rel.err.v
2063
2064 close unit 3
2065
2066 integrator.v = 'runge.kutta.r'
2067
2068
2069 open 3 for input, name is "seed.dat"    ''entries to seed.v
2070 use 3 for input
2071
2072 release seed.v(*)
2073 read no.streams
2074 start new record
2075 reserve seed.v(*) as no.streams
2076 for i = 1 to no.streams
2077 do

```

```

2078         read seed.v(i)
2079         start new record
2080     loop
2081
2082     close unit 3
2083
2084 end '' of get.sim
2085
2086
2087 function get.table      '' <f>18
2088 '' <s> get.table
2089 '' reads table of average arrivals in interval into an array
2090 '' of cumulative arrivals by interval
2091
2092     define table as a double, 2-dim array
2093     define temp, cum as double variables
2094     define i, n as integer variables
2095
2096     open unit 3 for input, name is "wcrate.dat"
2097     use 3 for input
2098
2099     read n
2100     reserve table(*,*) as n by 2
2101     for i = 1 to n
2102     do
2103         start new record
2104         read table(i, 1), temp
2105         add temp to cum
2106         table(i, 2) = cum
2107     loop
2108     close unit 3
2109
2110     return with table(*,*)
2111
2112 end '' get.table
2113
2114
2115 function get.travel.time(here, there, type, stream)      '' <f>12
2116 '' <s> get.travel.time
2117 '' returns travel time from here to there, constrained to be > 1 minute,
2118 '' with shape parameter adjusted to make final result approx gamma(3)
2119
2120     define here, there, type, stream as integer variables
2121     define t, temp as double variables
2122
2123     select case type
2124
2125     case ground
2126         temp = transit.time(here, there)
2127         t = 1 + mygamma.f(temp - 1, 3 * ((temp - 1) / temp)**2, stream) +
2128             choke.f(here, there)
2129
2130     case air
2131         temp = flight.time(here, there)
2132         t = 1 + mygamma.f(temp - 1, 3 * ((temp - 1) / temp)**2, stream)
2133
2134     endselect
2135
2136     return with t
2137
2138 end '' of get.travel.time
2139
2140

```

```

2141 function go.green(h, n)      ''      <f>12
2142 '' <s> go.green
2143
2144     define h, n as integer variables
2145
2146     define result as an integer variable
2147
2148     if n <= capacity(h) - green.limit
2149         result = TRUE
2150     else
2151         result = FALSE
2152     endif
2153
2154     return with result
2155
2156 end '' of go.green
2157
2158
2159 event go.off.red given hsp      '' <f>12
2160 '' <s> go.off.red
2161
2162     define hsp as an integer variable
2163     define cn as a pointer variable
2164
2165     if (time.v - red.start(hsp)) * hours.v <= max.red
2166         if go.green(hsp, no.pts(hsp)) = FALSE
2167             '' schedule another check
2168             create a go.off.red called cn
2169             clear.notice(hsp) = cn
2170             schedule the go.off.red called cn giving hsp in min.red hours
2171             return
2172         endif
2173     endif
2174
2175     ''max red time up, or load has decreased now
2176     clear.notice(hsp) = NULL
2177     red(hsp) = FALSE
2178     remove hsp from the red.set
2179     file hsp in the green.set
2180     return
2181
2182 end '' of go.off.red
2183
2184
2185 function go.red(h, n)          ''      <f>12
2186 '' <s> go.red
2187
2188     define h, n as integer variables
2189
2190     define result as an integer variable
2191
2192     if can.divert(h) = TRUE and n.red.set < max.red.hosp and
2193         red.today(h) < reds.per.day and n >= capacity(h) + red.limit
2194         result = TRUE
2195     else
2196         result = FALSE
2197     endif
2198
2199     return with result
2200
2201 end '' of go.red
2202
2203

```

```

2204 routine init.accident(accd)      '' <f>18
2205 '' <s> init.accident
2206
2207     define accd as a pointer variable
2208
2209     define this.patient as a pointer variable
2210     define i as an integer variables
2211     define x as a double variable
2212
2213     acc.start.time(accd) = time.v
2214     pended(accd) = FALSE
2215     if random.f(22) <= tr.prop
2216         add 1 to acc.counter
2217         kind(accd) = trauma
2218     else
2219         add 1 to med.counter
2220         kind(accd) = medical
2221     endif
2222     if random.f(28) > .3
2223         blunt(accd) = TRUE
2224         if kind(accd) = trauma
2225             add 1 to no.blunt
2226         endif
2227     else
2228         blunt(accd) = FALSE
2229     endif
2230
2231     x = random.f(8)
2232     for each node with prop.accs(node, 2) > x
2233         find the first case
2234         site(accd) = node
2235         file accd in the active.set
2236
2237         '' create victims for this accident
2238         if kind(accd) = medical      ''dont model medical patients
2239             add 1 to m.patient.counter
2240             add 1 to patient.counter
2241             no.victims(accd) = 1
2242         else
2243             no.victims(accd) = poisson.f(1.5, 9)
2244             for i = 1 to no.victims(accd)
2245                 do
2246                     add 1 to patient.counter
2247                     add 1 to t.patient.counter
2248                     create a patient called this.patient
2249                     pt.id(this.patient) = patient.counter
2250                     acc(this.patient) = accd
2251                     call init.pt giving this.patient
2252                     file this.patient in the patient.set
2253                     file this.patient in the acc.patient.set
2254                     activate the patient called this.patient now
2255                 loop
2256             endif
2257
2258 end '' of init.accident
2259
2260 routine init.pt given pt          '' <f>12
2261 '' <s> init.pt
2262
2263     define pt as a pointer variable
2264
2265     define rem.iss, rem.iss.1, t1, t2 as double variables

```

```

2267
2268 phase(pt) = waiting
2269 blunt(pt) = blunt(acc(pt))
2270 injury.loc(pt) = site(acc(pt))
2271 t1 = random.f(25)
2272 t2 = random.f(26)
2273
2274 '' beta args from fit to Bakers data w/ percentile matching checked
2275 '' on Baker and on Mackenzie
2276
2277 r.cts(pt) = 4
2278 h.cts(pt) = 4
2279 n.cts(pt) = 4
2280 iss(pt) = int.f(.5 + 75.0 * mybeta.f(1.390, 9.632, 10))
2281 rem.iss = iss(pt)
2282
2283 if iss(pt) >= 10          ''severely injured
2284     '' calculate CNS component -- data from Baxt
2285     if t1 < .40          ''40% of these have CNS injury
2286         add -2 to r.cts(pt)
2287         if iss(pt) < 15
2288             add -2 to n.cts(pt)
2289             rem.iss = iss(pt) / 2
2290         else
2291             if iss(pt) < 32
2292                 add -3 to n.cts(pt)
2293                 rem.iss = iss(pt) / 3
2294             else
2295                 add -4 to n.cts(pt)
2296                 rem.iss = iss(pt) / 5
2297             endif
2298         endif
2299     endif
2300
2301 else                    ''not severely injured
2302     if t1 < .20          ''only 20% have CNS component
2303         add -1 to n.cts(pt)
2304         add -1 to r.cts(pt)
2305         rem.iss = iss(pt) / 2
2306     endif
2307 endif
2308
2309 '' now partition hemorrhage and respiratory injury
2310 if t2 < .90            ''almost all injuries have some bleeding
2311     if t2 < .09          ''10% will have some resp injury also
2312         rem.iss.1 = .9 * rem.iss
2313     else
2314         rem.iss.1 = rem.iss
2315     endif
2316     '' scale bleeding rate from 4 to 225 ml / min, w/ 50% < 34
2317     br.0(pt) = 4 + 3 * rem.iss.1 ''br in ml/min
2318
2319     br.0(pt) = br.0(pt) * hours.v * minutes.v
2320     rem.iss = rem.iss - rem.iss.1
2321
2322 endif
2323
2324 '' what remains is respiratory component
2325 r.cts(pt) = max.f(0, r.cts(pt) - int.f(.5 + sqrt.f(rem.iss)))
2326 O2.sat(pt) = .80 + .05 * r.cts(pt)
2327
2328 cts(pt) = cts.f(pt)
2329

```



```

2330 end '' of init.pt
2331
2332
2333 routine initialize      '' <f>18
2334 '' <s> initialize
2335
2336     '' get time series of runs
2337     cum.events(*, *) = get.table
2338
2339     '' get general simulation data
2340     call get.sim
2341
2342     '' get transportation network
2343     call get.net
2344
2345     '' get hospitals
2346     call get.hosp
2347
2348     '' get ambulances
2349     call get.ems
2350
2351     '' get ambulance call list
2352     call get.list("call.dat", 'read.call.list', 'build.call.list',
2353     'write.call.list')
2354
2355     '' get hospital preference list
2356     call get.list("go.dat", 'read.hosp.list', 'build.hosp.list',
2357     'write.hosp.list')
2358
2359     '' output net, call list, etc for confirmation
2360     call print.net
2361
2362     '' initialize distribution of accidents across nodes
2363     prop.accs(*, *) = get.accs
2364
2365 end '' of initialize
2366
2367
2368 function laplace.f(location, scale, stream)
2369 '' returns a laplace variate -- for use in the gamma function
2370
2371     define location, scale as double variables
2372     define stream as an integer variable
2373
2374     define y, u as double variables
2375
2376     let u = random.f(stream)
2377     if u < .5
2378         let y = log.e.f(2 * u) * scale
2379     else
2380         let y = -log.e.f(2.0 - 2 * u) * scale
2381     endif
2382
2383     return with location + y
2384
2385 end ''laplace.f
2386
2387
2388 function lin.int.f(index, table)      '' <f>18
2389 '' <s> lin.int.f
2390 '' performs linear interpolation as follows:
2391 '' given a value b=B and a table(a, b), where b is a cumulative value,
2392 '' returns an interpolated value for a

```

```

2393
2394     define index as a double variable
2395     define table as a double, 2-dim array
2396     define x, slope, intercept as double variables
2397     define i, maxi as integer variables
2398
2399     maxi = dim.f(table(*,*))
2400
2401     for i = 1 to maxi
2402         with table(i, 2) >= index
2403         find the first case
2404         x = table(i, 2)
2405         slope = (x - table(i - 1, 2)) / (table(i, 1) - table(i - 1, 1))
2406         intercept = x - table(i, 1) * slope
2407
2408         return ((index - intercept) / slope)
2409
2410 end '' lin.int.f
2411
2412
2413 function living(pt) '' <f>12
2414 '' <s> living
2415
2416     define pt as a pointer variable
2417
2418     if O2.delivery(pt) le croak
2419         return with FALSE
2420     endif
2421     return with TRUE
2422
2423 end '' of living
2424
2425
2426 function mybeta.f(a, b, stream) '' <f>12
2427 '' <s> mybeta.f
2428
2429     define a, b as double variables
2430     define stream as an integer variable
2431
2432     define x as a double variable
2433
2434     let x = mygamma.f(a, a, stream)
2435     let x = x / (x + mygamma.f(b, b, stream))
2436
2437     return with x
2438
2439 end ''mybeta.f
2440
2441
2442 function mygamma.f(mean, a, stream) '' <f>12
2443 '' <s> mygamma.f
2444
2445 '' function to replace error prone gamma.f function
2446
2447 '' for order > 1
2448 '' taken from Tadikamalla as reported in Bratley B, Fox BL, Schrage LE:
2449 ''     A Guide to Simulation. New York, Springer-Verlag, 1987.
2450
2451 '' for order <= 1, taken from Ahrens, reported in same source
2452
2453     define mean, a as double variables
2454     define stream as an integer variable
2455

```

```

2456 define b, temp, y, u, g, scale, u1, u2, p, q as double variables
2457 define .terminated. to mean 1 = 2
2458
2459 '' check for bad arg's
2460 if mean <= 0.0
2461     let err.f = 145
2462 endif
2463 if a <= 0.0
2464     let err.f = 146
2465 endif
2466
2467 let b = mean / a
2468 let temp = a - 1.0
2469
2470 '' first generate gamma variate for order a, mean a (b==1)
2471 if a <= 1.0
2472     until .terminated.
2473     do
2474         let u1 = random.f(stream)
2475         let g = (exp.c + a) / exp.c
2476         let p = u1 * g
2477         let u2 = random.f(stream)
2478         if p > 1.0
2479             y = -log.e.f((g - p) / a)
2480             q = temp * log.e.f(y)
2481         else
2482             y = exp.f(log.e.f(p) / a)
2483             q = - y
2484         endif
2485         if log.e.f(u2) <= q
2486             leave
2487         endif
2488     loop
2489 else '' a > 1.0
2490     until .terminated.
2491     do
2492         until .terminated.
2493         do
2494             let scale = 1.0 + sqrt.f(4.0 * a - 3.0) / 2.0
2495             let y = laplace.f(temp, scale, stream)
2496             if y >= 0.0
2497                 leave '' inner loop
2498             endif
2499         loop
2500         let u = random.f(stream)
2501         let g = (((scale - 1.0) * y) / (scale * temp)) ** temp *
2502             exp.f(-y + (abs.f(y - temp) + temp * (scale + 1.0)) / scale)
2503         if u <= g
2504             leave '' outer loop
2505         endif
2506     loop
2507 endif
2508
2509 '' now y is gamma a=a, b=1, so scale to a=a, b=b
2510 return with y * b
2511
2512 end ''mygamma.f
2513
2514
2515 left routine no.pts given hospital '' <f>12
2516 '' <s> no.pts
2517 '' used to ACCUMULATE pt stats by hospital -- could have been done
2518 '' by compiler, but now used to check red status dynamically

```

```

2519
2520     define hospital as an integer variable
2521
2522     define n as an integer variable
2523     define inc as a double variable
2524
2525     enter with n                ''new value for no.pts
2526     inc = time.v - update.time(hospital)
2527
2528     add (no.pts(hospital) * inc) to r.accum.pts(hospital)
2529     add (full(hospital) * inc) to r.accum.cap(hospital)
2530     if n >= capacity(hospital)
2531         full(hospital) = TRUE
2532     else
2533         full(hospital) = FALSE
2534     endif
2535     call check.red giving hospital and n
2536     r.max.pts(hospital) = max.f(r.max.pts(hospital), n)
2537     update.time(hospital) = time.v
2538
2539     move from n
2540     return
2541
2542 end '' of no.pts
2543
2544
2545 function nsp.f(table, stream)    '' <f>24
2546     '' <s> nsp.f
2547     '' returns time til the next event for a non-stationary Poisson arrival
2548     '' process, given a table of times and cumulative mean arrivals
2549     '' The arrival times are assumed to wrap around when the end of the
2550     '' table is reached. Caller is responsible for ensuring that start of
2551     '' simulation is at the time represented by the first entry in the table.
2552
2553     '' algorithm from Çinlar, E. "Introduction to Stochastic Processes,"
2554     '' Prentice-Hall, Englewood Cliffs, NJ, 1975, pp 94-101
2555
2556     define table as a double, 2-dim array
2557     define stream as an integer variable
2558
2559     define maxi, x as integer variables
2560     define index, max.arr, y, z as double variables
2561
2562     '' define tprime, last.time as saved, double variables
2563     '' these made global to allow full reset
2564
2565     maxi = dim.f(table(*,*))
2566     max.arr = table(maxi, 2)
2567
2568     nsp.tprime = nsp.tprime - log.e.f(random.f(stream))
2569
2570     x = trunc.f(nsp.tprime / max.arr)    ''number of wraps around end of table
2571     index = mod.f(nsp.tprime, max.arr)  ''offset into table
2572
2573     y = lin.int.f(index, table(*,*)) + x * table(maxi, 1) ''time of next event
2574     z = y - nsp.last.time
2575     nsp.last.time = y
2576     return (z)
2577
2578 end '' nsp.f
2579
2580
2581 routine pass.time given pt    '' <f>12

```

```

2582 '' <s> pass.time
2583 '' models passage of time for alive patients, and for pts who have
2584 '' become terminal but will still receive an attempt at resuscitation
2585
2586 define pt as a pointer variable
2587
2588 define next.phase as an integer variable
2589
2590 select case phase(pt)
2591     case pvt.trav
2592         next.phase = resus
2593
2594     case scene.rx
2595         next.phase = en.route
2596
2597     case en.route
2598         next.phase = resus
2599
2600     case resus
2601         next.phase = done.resus
2602 endselect
2603
2604 until phase(pt) = next.phase
2605 do
2606     work continuously evaluating 'bleed', testing 'done',
2607     updating 'update'
2608
2609     '' if pt dies in the work continuously statement, record
2610     '' appropriate info and continue until the next phase change
2611     if condition(pt) = alive ''if just now died
2612         if living(pt) = FALSE
2613             if DEBUG
2614                 write pt.id(pt), hour.f(time.v), minute.f(time.v),
2615                 phase(pt) as
2616                 "pt ", i 4, " arrests at ", i 2, ":", i 2,
2617                 " in phase", i 2, /
2618             endif
2619             tod(pt) = (time.v - injury.time(pt)) * hours.v * minutes.v
2620             add 1 to no.deaths
2621             condition(pt) = dead
2622         endif
2623     endif
2624     if phase(pt) = pvt.trav
2625         '' NB -- this condition must agree with termination condition in
2626         '' the done function
2627         if time.v > pvt.tr.arrive
2628             phase(pt) = next.phase
2629         endif
2630     endif
2631     if phase(pt) = resus
2632         '' same type problem as above
2633         if time.v > injury.time(pt) + (wait.time(pt) + scene.time(pt) +
2634         transp.time(pt) + resus.time(pt)) / hours.v / minutes.v
2635             phase(pt) = next.phase
2636         endif
2637     endif
2638 loop
2639
2640 cts(pt) = cts.f(pt)
2641 change.flag(pt) = FALSE
2642
2643 end '' of pass.time
2644

```

```

2645
2646 process patient      '' <f>18
2647 '' <s> patient
2648
2649     define self as a pointer variable
2650
2651     define t as a double variable
2652
2653     self = patient
2654     change.flag(self) = FALSE
2655     injury.time(self) = time.v
2656     sbp(self) = sbp.0
2657     blood.volume(self) = bv.0
2658     hct(self) = hct.0
2659     rbc.mass(self) = .01 * hct(self) * blood.volume(self)
2660     O2.delivery(self) = sbp(self) * hct(self) * O2.sat(self)
2661
2662     bleeding.rate(self) = br.0(self)
2663     condition(self) = alive
2664
2665     resus.time(self) = 1 + mygamma.f(t.resus - 1,
2666         3 * ((t.resus - 1) / t.resus)**2, 11)
2667     tx.time(self) = exponential.f(t.tx, 24)
2668     t = time.v
2669
2670     '' bleed till arrival
2671     work continuously evaluating 'bleed', testing 'done', updating 'update'
2672     change.flag(self) = FALSE
2673     cts(self) = cts.f(self)
2674
2675     if living(self) = FALSE
2676         condition(self) = dead
2677         tod(self) = (time.v - injury.time(self)) * hours.v * minutes.v
2678         call pt.report giving self
2679         remove self from acc.patient.set(acc(self))
2680         remove self from the patient.set
2681         add 1 to no.deaths
2682         if DEBUG
2683             write pt.id(self), hour.f(time.v), minute.f(time.v), phase(self) as
2684             "pt ", i 4, " arrests at ", i 2, ":", i 2, " in phase", i 2, /
2685         endif
2686         return
2687     endif
2688     wait.time(self) = (time.v - t) * hours.v * minutes.v
2689     t = time.v
2690
2691     if phase(self) = pvt.trav
2692         call pass.time giving self
2693         remove self from the pvt.tr.set
2694         file self in resus.patient.set(hosp(self))
2695         phase(self) = resus
2696         transp.time(self) = (time.v - t) * hours.v * minutes.v
2697         t = time.v
2698     else
2699         '' ambulance arrival
2700         rescue.bp(self) = sbp(self)
2701         call pass.time giving self
2702         scene.time(self) = (time.v - t) * hours.v * minutes.v
2703         t = time.v
2704
2705         '' enroute
2706         call pass.time giving self
2707         transp.time(self) = (time.v - t) * hours.v * minutes.v

```

```

2708         t = time.v
2709     endif
2710
2711     '' hospital arrival
2712     hosp.bp(self) = sbp(self)
2713     trf.start.time(self) = time.v + resus.time(self) / hours.v / minutes.v / 2
2714     schedule a resp.support(self) in resus.time(self) / 3 minutes
2715     trf.rate(self) = .5 * iv.rate(self)
2716     no.pts(hosp(self)) = no.pts(hosp(self)) + 1
2717     call pass.time giving self
2718
2719
2720     if condition(self) = alive      '' if didn't die earlier
2721         if living(self) = FALSE
2722             condition(self) = dead
2723             add 1 to no.deaths
2724             if DEBUG
2725                 write pt.id(self), hour.f(time.v), minute.f(time.v) as
2726                 "pt ", i 4, " arrests at ", i 2, ":", i 2, " after resus", /
2727             endif
2728         else
2729             '' transfer to definitive care
2730             work tx.time(self) minutes
2731             def.bp(self) = sbp(self)
2732             cts(self) = cts.f(self)
2733         endif
2734     endif
2735
2736     '' done with this patient, for purposes of this model
2737
2738     remove self from the patient.set
2739     remove self from the resus.patient.set(hosp(self))
2740     add -1 to no.pts(hosp(self))
2741
2742     '' coefficients from MTOS, assuming age < 55
2743     if blunt(self) = TRUE
2744         prob.surv(self) = 1 /
2745         (1 + exp.f(-(-1.247 + .9544 * cts(self) -.0768 * iss(self))))
2746     else
2747         prob.surv(self) = 1 /
2748         (1 + exp.f(-(-.6029 + 1.1430 * cts(self) -.1516 * iss(self))))
2749     endif
2750
2751     add (1 - prob.surv) to est.deaths
2752     add (1 - prob.surv) to r.est.deaths
2753
2754     call pt.report giving self
2755
2756 end '' patient
2757
2758
2759 routine print.net      '' <f>12
2760 '' <s> print.net
2761 '' prints system information as defined by the data files for confirmation
2762
2763     define i as an integer variable
2764     define lvl as a text variable
2765     define item as a pointer variable
2766     define temp as a double variable
2767
2768     use 4 for output
2769     skip 1 line
2770     print 2 lines thus

```



```

2834         select case level(hospital.id(item))
2835             case level1
2836                 lvl = "level 1"
2837             case level2
2838                 lvl = "level 2"
2839             case level2a
2840                 lvl = "level 2a"
2841             case level3
2842                 lvl = "level 3"
2843         endselect
2844
2845         print 1 line with hosp.name(hospital.id(item)),
2846             lvl, node.name(hosp.base(hospital.id(item))),
2847             transit.time(node, hosp.base(hospital.id(item)))
2848         thus
2849         ***, *****, in ***, with mean travel time ****.** min
2850     loop
2851 loop
2852 skip 1 line
2853
2854 print 7 lines with major.cutoff, minor.cutoff, htol*100, atol*100 thus
2855 Triage rule:
2856     Champ TS <= *.* goes to level 1, > *.* may go to level 3
2857
2858 Travel times exceeding minimum time by less than tolerance included in
2859     routine dispatch lists
2860 hospital choice tolerance **%
2861 ambulance choice tolerance **%
2862 skip 1 line
2863 use 6 for output
2864
2865 end '' of print.net
2866
2867
2868 routine pt.report given pt      '' <f>12
2869 '' <s> pt.report
2870
2871     define pt as a pointer variable
2872
2873     use 8 for output
2874
2875     write run, pt.id(pt), blunt(pt), transp.mode(pt), iss(pt),
2876         injury.time(pt), wait.time(pt), scene.time(pt), transp.time(pt),
2877         resus.time(pt), tx.time(pt), tod(pt), rescue.bp(pt),
2878         hosp.bp(pt), def.bp(pt) as
2879         i 3, s 1, i 5, 2 i 3, i 5, d(22, 15), 9 d(10, 3), +
2880
2881         write
2882         prob.surv(pt), br.0(pt) / hours.v / minutes.v, cts(pt),
2883         condition(pt), injury.loc(pt), hosp(pt), need.helo(pt),
2884         got.helo(pt) as
2885         3 d(10, 3), 5 i 4, /
2886
2887     use 6 for output
2888
2889 end '' of pt.report
2890
2891
2892 routine pvt.travel(acc)      '' <f>12
2893 '' <s> pvt.travel
2894
2895     define acc as a pointer variable
2896

```

```

2897     define pt, item as pointer variables
2898     define temp, t, limit as double variables
2899     define hsp, rtot, cumtot as integer variables
2900
2901     '' find nearest hospital, regardless of status
2902     limit = random.f(16)
2903     rtot = 0
2904     for each item in hosp.list(site(acc))
2905         compute cumtot as the sum of hosp.volume(hospital.id(item))
2906     for each item in hosp.list(site(acc))
2907     do
2908         add hosp.volume(hospital.id(item)) to rtot
2909         if rtot / cumtot >= limit
2910             hsp = hospital.id(item)
2911             leave
2912         endif
2913     loop
2914
2915     until acc.patient.set(acc) is empty
2916     do
2917         remove first pt from acc.patient.set(acc)
2918         t = 1.2 * transit.time(site(acc), hosp.base(hsp))
2919         temp = 1 + mygamma.f(t - 1, 3 * ((t - 1) / t)**2, 20) +
2920             choke.f(site(acc), hosp.base(hsp))
2921         '' pvt travel time 20% more than ambulance
2922         pvt.tr.arrive(pt) = time.v + temp / minutes.v / hours.v
2923         phase(pt) = pvt.trav
2924         change.flag(pt) = TRUE
2925         hosp(pt) = hsp
2926         transp.mode(pt) = private
2927         file pt in the pvt.tr.set
2928     loop
2929
2930 end '' of pvt.travel
2931
2932
2933 routine read.call.list '' <f>18
2934 '' <s> read.call.list
2935 '' unit 3 is already open on entry
2936
2937     define here as an integer variable
2938
2939     while data is not ended
2940     do
2941         read here
2942         while card is not new
2943         do
2944             create a call.item
2945             read ambulance.id(call.item)
2946             file this call.item in call.list(here)
2947         loop
2948     loop
2949     close unit 3
2950
2951 end '' of read.call.list
2952
2953
2954 routine read.hosp.list '' <f>12
2955 '' <s> read.hosp.list
2956 '' unit 3 is already open on entry
2957
2958     define here as an integer variable
2959

```

```

2960     while data is not ended
2961     do
2962         read here
2963         while card is not new
2964         do
2965             create a go.item
2966             read hospital.id(go.item)
2967             read hosp.level(go.item)
2968             file this go.item in hosp.list(here)
2969         loop
2970     loop
2971     close unit 3
2972
2973 end '' of read.call.list
2974
2975
2976 routine recall given num, acc      '' <f>18
2977 ''<s> recall
2978 '' recall num ambulances starting from the end of the
2979 '' amb.set (farthest from acc)
2980
2981     define num as an integer variable
2982     define acc as a pointer variable
2983
2984     define i, amb as integer variables
2985     define arun as a pointer variable
2986
2987     i = 0
2988     for each amb in the amb.set(acc) in reverse order, with type(amb)=ground
2989     do
2990         if amb is in the to.scene.set
2991             add 1 to i
2992             remove amb from the amb.set(acc)
2993             remove amb from the to.scene.set
2994             arun = amb.run(amb)
2995             interrupt the ambulance.run called arun
2996             status(arun) = not.working
2997             cur.location(amb) = find.loc(amb)
2998             time.a(arun) = 0
2999             resume the ambulance.run called arun
3000             if DEBUG
3001                 write amb.name(amb), hour.f(time.v), minute.f(time.v) as
3002                 t 3, " recalled at ", i 2, ":", i 2, /
3003             endif
3004         endif
3005         if i = num
3006             leave
3007         endif
3008     loop
3009
3010 end '' of recall
3011
3012
3013 event resp.support given pt      '' <f>12
3014 '' <s> resp.support
3015 '' event to represent provision of O2, intubations, chest decompression, etc
3016
3017     define pt as a pointer variable
3018
3019     r.cts(pt) = 4.0 - r.cts(pt) / 2
3020     O2.sat(pt) = .80 + .05 * r.cts(pt)
3021     cts(pt) = cts.f(pt)
3022

```

```

3023 end '' of resp.support
3024
3025
3026 routine run.report      '' <f>12
3027 '' <s> run.report
3028
3029     define p as a double variable
3030
3031     if r.no.pended > 0
3032         p = r.mean.pending * tmp.dur * hours.v * minutes.v / r.no.pended
3033     else
3034         p = 0
3035     endif
3036
3037     use 4 for output
3038     print 10 lines with run, tmp.dur * hours.v,
3039         r.no.accs, r.no.t.pts, 100 * r.no.blunt / r.no.accs,
3040         1.0 - r.idle.amb / (n.ambulance - num.helo),
3041         1.0 - r.idle.helo / num.helo,
3042         r.no.divert, r.no.override, r.no.deaths,
3043         r.est.deaths thus
3044
3045 run **
3046     duration      ***.** hours      no.accs      ***      no. tr. pts  ***
3047     % blunt              **.**
3048     amb. utilization    ***.**
3049     helo utilization    ***.**
3050     diverts            ***
3051     overrides          ***
3052     deaths             ***
3053     est deaths         ***.**
3054
3055     print 4 lines with r.mean.pending, r.max.pending, r.no.pended, p
3056     thus
3057     queue for amb mean      ***.**      max      ***
3058     no.accs pended:        ****
3059     av pending time:      ***.** minutes
3060
3061     print 2 lines thus
3062     hospital utilization
3063     name      avg tr load      max tr load      reserve      % red time
3064     for each hospital
3065     do
3066         print 1 line with hosp.name(hospital), r.accum.pts(hospital) / tmp.dur,
3067             r.max.pts(hospital), 1 - r.accum.cap(hospital) / tmp.dur,
3068             r.mean.red(hospital) * 100 thus
3069     ***      ***.**      ***      *.***      ***.*
3070     loop
3071     skip 1 line
3072
3073     use 7 for output
3074     ''run, duration, midpoint, no.accs, no.blunt, no.pts, amb utilization, hosp
3075     ''utilization, reserve, diverts, overrides, deaths, est.deaths
3076     write run, tmp.dur, time.v - tmp.dur / 2.0, r.no.accs, r.no.meds,
3077         r.no.blunt, r.no.t.pts, 1.0 - r.idle.amb / (n.ambulance - num.helo),
3078         1.0 - r.idle.helo / num.helo, r.no.divert, r.no.override, r.no.deaths
3079     as
3080     i 5, 2 d(12, 5), 4 i 7, 2 d(10, 5), 3 i 5, +
3081     write r.est.deaths, r.no.pended, p as
3082     d(8, 3), i 7, d(8, 3), +
3083
3084     '' repeat the loop so variables are grouped (easier to read in SYSTAT)
3085     for each hospital

```

```

3086     do
3087         write r.accum.pts(hospital) / tmp.dur as
3088             d(12, 5), +
3089     loop
3090     for each hospital
3091     do
3092         write 1 - r.accum.cap(hospital) / tmp.dur as
3093             d(10, 5), +
3094     loop
3095     for each hospital
3096     do
3097         write r.mean.red(hospital) as
3098             d(10, 5), +
3099     loop
3100     write as /
3101
3102     use 6 for output
3103
3104 end '' run.report
3105
3106 routine tr.check.in.acc given acc      '' <f>12
3107 '' <s> tr.check.in.acc
3108
3109     define acc as a pointer variable
3110     define t as a text variable
3111
3112     if kind(acc) = trauma
3113         t = "acc"
3114     else
3115         t = "med"
3116     endif
3117     write acc.id(acc), t, node.name(site(acc)),
3118         hour.f(time.v), minute.f(time.v) as
3119         "event ", i 5, " (", t 3, ") occurs in ", t 3, " at ", i 2, ":", i 2, /
3120
3121 end '' of tr.check.in.acc
3122
3123
3124 routine tr.check.in.amb given amb      '' <f>12
3125 '' <s> tr.check.in.amb
3126
3127     define amb as an integer variable
3128
3129     write amb.name(amb), node.name(amb.base(amb)), hour.f(time.v),
3130         minute.f(time.v) as
3131         " ", t 3, " at base (", t 3, ") at ", i 2, ":", i 2, /
3132
3133 end '' of tr.check.in.amb
3134
3135
3136 routine tr.check.in.pt given pt and acc '' <f>12
3137 '' <s> tr.check.in.pt
3138
3139     define pt, acc as pointer variables
3140
3141     write acc.id(acc), pt.id(pt) as
3142         "acc ", i 5, " has pt ", i 5, /
3143
3144 end '' tr.check.in.pt
3145
3146
3147 routine tr.check.out.acc given acc      '' <f>12

```

```

3149 '' <s> tr.check.out.acc
3150
3151     define acc as a pointer variable
3152     define t as a text variable
3153
3154     if kind(acc) = trauma
3155         t = "acc"
3156     else
3157         t = "med"
3158     endif
3159
3160     write acc.id(acc), t, hour.f(time.v), minute.f(time.v) as
3161         "event ", i 5, " (" , t 3, " ) done at ", i 2, ":", i 2, /
3162
3163 end '' of tr.check.out.acc
3164
3165
3166 routine tr.check.out.pt given pt      '' <f>12
3167 '' <s> tr.check.out.pt
3168
3169     define pt as a pointer variable
3170
3171     write pt.id(pt), hour.f(time.v), minute.f(time.v), n.patient.set as
3172         "          pt ", i 5, " finished at ", i 2, ":", i 2,
3173         " leaving ", i 3, " in system", /
3174
3175 end '' of tr.check.out.pt
3176
3177
3178 routine tr.deliver.pt given amb      '' <f>12
3179 '' <s> tr.deliver.pt
3180
3181     define amb as an integer variable
3182
3183     write amb.name(amb), hosp.name(hosp(amb.run(amb))),
3184         hour.f(time.v), minute.f(time.v)
3185         as "          ", t 3, " at ", t 3, " at ",
3186         i 2, ":", i 2, /
3187
3188 end '' of tr.deliver.pt
3189
3190
3191 routine tr.enroute.hosp(amb)          '' <f>12
3192 '' <s> tr.enroute.hosp
3193
3194     define amb as an integer variable
3195     define n as an integer variable
3196
3197     if kind(amb.run(amb)) = trauma
3198         n = n.amb.patient.set(amb.run(amb))
3199     else
3200         n = 1
3201     endif
3202
3203     write amb.name(amb), hosp.name(hosp(amb.run(amb))), hour.f(time.v),
3204         minute.f(time.v), n as
3205         "          ", t 3, " enroute to ", t 3, " at ",
3206         i 2, ":", i 2, " with ", i 2, " pts", /
3207
3208 end '' of tr.enroute.hosp
3209
3210
3211 routine tr.go.green(hsp)              '' <f>12

```

```

3212 '' <s> tr.go.green
3213
3214     define hsp as an integer variable
3215
3216     write hosp.name(hsp), hour.f(time.v), minute.f(time.v) as
3217     t 3, " went green at ", i 2, ":", i 2, /
3218
3219 end '' of tr.go.green
3220
3221 routine tr.go.red(hsp)          '' <f>12
3222 '' <s> tr.go.red
3223
3224     define hsp as an integer variable
3225
3226     write hosp.name(hsp), hour.f(time.v), minute.f(time.v) as
3227     t 3, " went red at ", i 2, ":", i 2, /
3228
3229 end '' of tr.go.red
3230
3231
3232 routine tr.on.scene given amb    '' <f>12
3233 '' <s> tr.on.scene
3234
3235     define amb as an integer variable
3236     define t as a text variable
3237
3238     if kind(amb.run(amb)) = trauma
3239         t = "acc"
3240     else
3241         t = "med"
3242     endif
3243
3244     write amb.name(amb), t, acc.id(acc(amb.run(amb))),
3245     node.name(cur.location(amb)), hour.f(time.v), minute.f(time.v) as
3246     " ", t 3, " on scene at ", t 3, s 1, i 5,
3247     " in ", t 3, " at ", i 2, ":", i 2, /
3248
3249 end '' of tr.on.scene
3250
3251
3252 routine tr.pickup.pt given pt and amb.run    '' <f>12
3253 '' <s> tr.pickup.pt
3254
3255     define pt, amb.run as pointer variables
3256
3257     if kind(amb.run) = trauma
3258         write amb.name(ambulance.id(amb.run)), pt.id(pt),
3259         cts(pt), hour.f(time.v), minute.f(time.v) as
3260         " ", t 3, " treating pt ", i 5, " ", cts = " ", d(3, 1),
3261         " at ", i 2, ":", i 2, /
3262     else
3263         write amb.name(ambulance.id(amb.run)),
3264         hour.f(time.v), minute.f(time.v) as
3265         " ", t 3, " treating med pt at ",
3266         i 2, ":", i 2, /
3267     endif
3268
3269 end '' of tr.pickup.pt
3270
3271
3272 routine tr.pvt.tr.pt given pt    '' <f>12
3273 '' <s> tr.pvt.tr.pt
3274

```

```

3275
3276     define pt as a pointer variable
3277
3278     write pt.id(pt), hosp.name(hosp(pt)),
3279         hour.f(time.v), minute.f(time.v) as
3280         "      pt ", i 5, " heading to ", t 3, " at ", i 2, ":", i 2, /
3281
3282 end '' of tr.pvt.tr.pt
3283
3284
3285 routine tr.resus.pt given pt and hosp '' <f>12
3286 '' <s> tr.resus.pt
3287
3288     define pt as a pointer variable
3289     define hosp as an integer variable
3290
3291     write pt.id(pt), hosp.name(hosp), weekday.f(time.v), hour.f(time.v) and
3292         minute.f(time.v) as
3293         "      pt ", i 5, " at ", t 3, " on day", i 2, " at ",
3294         i 2, ":", i 2, /
3295
3296 end '' of tr.resus.pt
3297
3298
3299 routine tr.send.amb given amb '' <f>12
3300 '' <s> tr.send.amb
3301
3302     define amb as an integer variable
3303     define t as a text variable
3304
3305     if kind(amb.run(amb)) = trauma
3306         t = "acc"
3307     else
3308         t = "med"
3309     endif
3310
3311     write amb.name(amb), t, acc.id(acc(amb.run(amb))),
3312         hour.f(time.v), minute.f(time.v)
3313         as "      ", t 3, " enroute to ", t 3, s 1, i 5, " at ",
3314         i 2, ":", i 2, /
3315
3316 end '' of tr.send.amb
3317
3318
3319 routine tr.stack.acc given acc '' <f>12
3320 '' <s> tr.stack.acc
3321
3322     define acc as a pointer variable
3323     define t as a text variable
3324
3325     if kind(acc) = trauma
3326         t = "acc"
3327     else
3328         t = "med"
3329     endif
3330
3331     write t, acc.id(acc) as
3332         t 3, s 1, i 5, " awaiting service", /
3333
3334 end '' tr.stack.acc
3335
3336
3337 routine tr.to.home given amb '' <f>12

```



```

3338 '' <s> tr.to.home
3339
3340     define amb as an integer variable
3341
3342     write amb.name(amb), hour.f(time.v), minute.f(time.v)
3343         as " ", t 3, " heading home at ",
3344         i 2, ":", i 2, /
3345
3346 end '' of tr.to.home
3347
3348
3349 routine tr.unstack.acc given acc      '' <f>12
3350 '' <s> tr.unstack.acc
3351
3352     define acc as a pointer variable
3353     define t as a text variable
3354
3355     if kind(acc) = trauma
3356         t = "acc"
3357     else
3358         t = "med"
3359     endif
3360
3361     write t, acc.id(acc), hour.f(time.v), minute.f(time.v)
3362         as "pending ", t 3, s 1, i 5, " served at ", i 2, ":", i 2, /
3363
3364 end '' of tr.unstack.acc
3365
3366
3367 routine travel(from, to, amb)      '' <f>12
3368 '' <s> travel
3369
3370     define from, to, amb as integer variables
3371
3372     remove amb from node.amb.set(from)
3373     work travel.time(amb) minutes
3374     file amb in node.amb.set(to)
3375     cur.location(amb) = to
3376
3377 end '' of travel
3378
3379
3380 routine update(patient)      '' <f>12
3381 '' <s> update
3382 '' these statements are required for changes in continuous simulation
3383 '' variables to be visible outside the integrator routine
3384
3385     define patient as a pointer variable
3386
3387     blood.volume(patient) = blood.volume(patient)
3388     d.blood.volume(patient) = d.blood.volume(patient)
3389
3390     rbc.mass(patient) = rbc.mass(patient)
3391     d.rbc.mass(patient) = d.rbc.mass(patient)
3392     sbp(patient) = sbp(patient)
3393     d.sbp(patient) = d.sbp(patient)
3394     bleeding.rate(patient) = bleeding.rate(patient)
3395     d.bleeding.rate(patient) = d.bleeding.rate(patient)
3396     hct(patient) = hct(patient)
3397     d.hct(patient) = d.hct(patient)
3398     O2.delivery(patient) = O2.delivery(patient)
3399     d.O2.delivery(patient) = d.O2.delivery(patient)
3400

```

```

3401 end '' of update
3402
3403
3404 routine write.call.list '' <f>12
3405 '' <s> write.call.list
3406
3407     define item as a pointer variable
3408
3409     open unit 3 for output, name is "call.dat"
3410     use 3 for output
3411     for each node
3412     do
3413         write node as i 3, s 2, +
3414         for each item in call.list(node)
3415         do
3416             write ambulance.id(item) as i 3, s 2, +
3417             loop
3418             write as /
3419         loop
3420     close unit 3
3421
3422 end '' of write.call.list
3423
3424
3425 routine write.hosp.list '' <f>12
3426 '' <s> write.hosp.list
3427
3428     define item as a pointer variable
3429
3430     open unit 3 for output, name is "go.dat"
3431     use 3 for output
3432     for each node
3433     do
3434         write node as i 3, s 2, +
3435         for each item in hosp.list(node)
3436         do
3437             write hospital.id(item), hosp.level(item) as i 3, s 2, i 3, s 2, +
3438             loop
3439             write as /
3440         loop
3441     close unit 3
3442
3443 end '' of write.call.list

```

Appendix 2

Data Cross-Reference

NAME	TYPE (+ distinct usage)	MODE	ROUTINE	REFERENCES (= assigned)
A.DUPLICATE	Define to mean		PREAMBLE BUILD.ROUTE	1 1
ABS.ERR.V	Permanent attribute	Real	GET.SIM	1 =
ABS.F	Library routine		BLEED MYGAMMA.F	2 1
ACC	Temporary attribute	Pointer	PREAMBLE AMBULANCE.RUN ASSIGN.AMB CHECK.ACCIDENT INIT.ACCIDENT INIT.PT PATIENT TR.ON.SCENE TR.SEND.AMB	3 1 1 = 1 = 1 = 2 1 1 1
ACC.ARRIVE.TIME	Temporary attribute	Double	PREAMBLE DISPATCHER	2 2 =
ACC.COUNTER	Global variable	Integer	PREAMBLE MAIN INIT.ACCIDENT	2 1 1 =
ACC.END.TIME	Temporary attribute	Double	PREAMBLE ACCIDENT	2 1 =
ACC.ID	Temporary attribute	Integer	PREAMBLE DISPATCHER GENERATOR TR.CHECK.IN.ACC TR.CHECK.IN.PT TR.CHECK.OUT.ACC TR.ON.SCENE TR.SEND.AMB TR.STACK.ACC TR.UNSTACK.ACC	2 3 1 = 1 1 1 1 1 1 1
ACC.PATIENT.SET	Set		PREAMBLE ACCIDENT AMBULANCE.RUN CHECK.ACCIDENT GET.PATIENT INIT.ACCIDENT PATIENT PVT.TRAVEL	3 1 4 1 2 1 1 2

ACC.START.TIME	Temporary attribute	Double	PREAMBLE INIT.ACCIDENT	2 1 =
ACCIDENT	Process notice		PREAMBLE ACCIDENT DISPATCHER GENERATOR	1 1 3 2
	+ Global variable	Pointer	ACCIDENT DISPATCHER GENERATOR	1 2 = 4 =
	+ Implied subscript		INIT.ACCIDENT ACCIDENT	1 1
ACTIVE.SET	Set		PREAMBLE ACCIDENT INIT.ACCIDENT	4 1 1
ADJ.TIME.F	Routine	Double	PREAMBLE ADJ.TIME.F BEST.ROUTE GET.NET	1 1 1 1
AIR	Define to mean		PREAMBLE AMBULANCE.RUN ASSIGN.AMB DISPATCHER GET.AMB GET.EMS GET.TRAVEL.TIME	1 3 1 2 1 1 1
AIRSPEED	Global variable	Double	PREAMBLE FTIME.F GET.NET	1 1 2 =
ALARM.LAG	Global variable	Double	PREAMBLE ACCIDENT GET.SIM	1 1 1 =
ALIVE	Define to mean		PREAMBLE DONE PASS.TIME PATIENT	1 1 1 2
AMB.BASE	Permanent attribute	Integer	PREAMBLE DISPATCHER GET.AMB GET.EMS PRINT.NET TR.CHECK.IN.AMB	2 2 2 5 = 3 1
AMB.BASE.SET	Set		PREAMBLE BUILD.CALL.LIST GET.EMS	2 1 2
AMB.ID	Permanent attribute	Integer	PREAMBLE GET.EMS	2 1 =
AMB.NAME	Permanent attribute	Text	PREAMBLE DISPATCHER FIND.HOSP GET.EMS PRINT.NET RECALL	2 2 2 1 = 1 1

			TR.CHECK.IN.AMB	1
			TR.DELIVER.PT	1
			TR.ENROUTE.HOSP	1
			TR.ON.SCENE	1
			TR.PICKUP.PT	2
			TR.SEND.AMB	1
			TR.TO.HOME	1
AMB.PATIENT.SET	Set		PREAMBLE	3
			AMBULANCE.RUN	7
			DISPATCHER	2
			GET.PATIENT	1
AMB.RUN	Permanent attribute	Pointer	PREAMBLE	2
			ASSIGN.AMB	1 =
			DISPATCHER	6
			FIND.HOSP	2
			FIND.LOC	5
			RECALL	1
			TR.DELIVER.PT	1
			TR.ENROUTE.HOSP	3
			TR.ON.SCENE	2
			TR.SEND.AMB	2
AMB.SET	Set		PREAMBLE	2
			ASSIGN.AMB	1
			CHECK.ACCIDENT	1
			DISPATCHER	2
			RECALL	2
AMBULANCE	Permanent entity		PREAMBLE	1
			GET.EMS	1
	+ Global variable	Integer2	GET.EMS	12 =
AMBULANCE.ID	Temporary attribute	Integer	PREAMBLE	3
			AMBULANCE.RUN	1
			ASSIGN.AMB	1 =
			BUILD.CALL.LIST	2 =
			CHECK.ACCIDENT	3
			GET.AMB	5
			GET.PATIENT	3
			PRINT.NET	4
			READ.CALL.LIST	1 =
			TR.PICKUP.PT	2
			WRITE.CALL.LIST	1
AMBULANCE.RUN	Process notice		PREAMBLE	1
			AMBULANCE.RUN	3
			ASSIGN.AMB	2
			RECALL	2
	+ Global variable	Pointer	AMBULANCE.RUN	1
			ASSIGN.AMB	9 =
	+ Implied subscript		AMBULANCE.RUN	1
ARC	Temporary entity		PREAMBLE	1
			BUILD.ROUTE	1
			GET.NET	1
	+ Global variable	Pointer	BUILD.ROUTE	7 =
			GET.NET	5 =
			PRINT.NET	2 =
ARC.STATUS	Temporary attribute	Integer	PREAMBLE	2
			BUILD.ROUTE	3 =

ASSIGN.AMB	Routine		ASSIGN.AMB GET.AMB	1 1
ASSIGNMENT.SET	Set		PREAMBLE BUILD.CALL.LIST	2 9
AT.BASE	Define to mean		PREAMBLE AMBULANCE.RUN DISPATCHER	1 1 1
AT.BASE.SET	Set		PREAMBLE ASSIGN.AMB DISPATCHER GET.EMS	2 1 1 1
AT.HOSP	Define to mean		PREAMBLE AMBULANCE.RUN DISPATCHER	1 1 1
AT.HOSP.SET	Set		PREAMBLE AMBULANCE.RUN DISPATCHER	3 1 1
AT.SCENE	Define to mean		PREAMBLE AMBULANCE.RUN DISPATCHER	1 1 1
ATOL	Global variable	Double	PREAMBLE BUILD.CALL.LIST GET.SIM PRINT.NET	1 1 1 = 1
BEST.ROUTE	Routine	Double	PREAMBLE BEST.ROUTE GET.NET	1 1 1
BLEED	Routine		BLEED PASS.TIME PATIENT	1 1 1
BLEEDING.RATE	Temporary attribute	Double	PREAMBLE BLEED PATIENT UPDATE	2 4 1 = 1 =
BLOOD.VOLUME	Temporary attribute	Double	PREAMBLE BLEED PATIENT UPDATE	2 4 2 = 1 =
BLUNT	Temporary attribute	Integer	PREAMBLE INIT.ACCIDENT INIT.PT PATIENT PT.REPORT	3 2 = 1 = 1 1
BR.0	Temporary attribute	Double	PREAMBLE BLEED INIT.PT PATIENT PT.REPORT	2 1 2 = 1 1
BUILD.CALL.LIST	Routine		BUILD.CALL.LIST	1

			INITIALIZE	1
BUILD.HOSP.LIST	Routine		BUILD.HOSP.LIST INITIALIZE	1 1
BUILD.ROUTE	Routine		BEST.ROUTE BUILD.ROUTE	2 1
BV.0	Define to mean		PREAMBLE BLEED PATIENT	1 2 1
CALL.ITEM	Temporary entity		PREAMBLE BUILD.CALL.LIST READ.CALL.LIST	1 2 1
	+ Global variable	Pointer	BUILD.CALL.LIST READ.CALL.LIST	7 = 3 =
CALL.LIST	Set		PREAMBLE BUILD.CALL.LIST GET.AMB PRINT.NET READ.CALL.LIST WRITE.CALL.LIST	2 3 2 1 1 1
CALLS.PENDING	Define to mean		PREAMBLE DISPATCHER	1 1
CAN.DIVERT	Permanent attribute	Integer	PREAMBLE GET.HOSP GO.RED	2 1 = 1
CAPACITY	Permanent attribute	Integer	PREAMBLE GET.HOSP GO.GREEN GO.RED LEFT NO.PTS PRINT.NET	2 1 = 1 1 1 1
CH.CUM.WEIGHT	Temporary attribute	Double	PREAMBLE BEST.ROUTE	2 5 =
CHANGE.FLAG	Temporary attribute	Integer	PREAMBLE AMBULANCE.RUN DONE GET.PATIENT PASS.TIME PATIENT PVT.TRAVEL	2 2 = 1 1 = 1 = 2 = 1 =
CHECK.ACCIDENT	Routine		AMBULANCE.RUN CHECK.ACCIDENT	2 1
CHECK.RED	Routine		CHECK.RED LEFT NO.PTS	1 1
CHOKE.F	Routine	Double	PREAMBLE CHOKE.F GET.TRAVEL.TIME PVT.TRAVEL	1 1 1 1
CHOKE.PT.WT	Temporary attribute	Double	PREAMBLE BEST.ROUTE	2 2

			BUILD.ROUTE	1 =
			CHOKE.F	2
			GET.NET	1 =
CLEAR.NOTICE	Permanent attribute	Pointer	PREAMBLE	2
			CHECK.RED	3 =
			CLEAR.REDS	3 =
			GO.OFF.RED	2 =
CLEAR.REDS	Event notice		PREAMBLE	1
			MAIN	1
			CLEAR.REDS	2
			GENERATOR	1
	+ Global variable	Pointer	MAIN	1 =
			CLEAR.REDS	2 =
			GENERATOR	2
CLEAR.TIME	Global variable	Double	PREAMBLE	1
			MAIN	1
			GENERATOR	1 =
			GET.SIM	1 =
CONDITION	Temporary attribute	Integer	PREAMBLE	2
			BLEED	1
			DONE	1
			PASS.TIME	2 =
			PATIENT	4 =
			PT.REPORT	1
COS.F	Library routine		FTIME.F	1
CROAK	Define to mean		PREAMBLE	1
			LIVING	1
CTS	Temporary attribute	Double	PREAMBLE	2
			AMBULANCE.RUN	1
			CTS.F	2 =
			GET.LEVEL	2
			INIT.PT	1 =
			PASS.TIME	1 =
			PATIENT	4 =
			PT.REPORT	1
			RESP.SUPPORT	1 =
			TR.PICKUP.PT	1
CTS.F	Routine	Double	PREAMBLE	2
			ACCIDENT	1
			CTS.F	1
			GET.PATIENT	1
			INIT.PT	1
			PASS.TIME	1
			PATIENT	2
			RESP.SUPPORT	1
CUM.EVENTS	Global variable	Double	PREAMBLE	1
			GENERATOR	2
			INITIALIZE	1 =
CUM.WEIGHT	Temporary attribute	Double	PREAMBLE	2
			BEST.ROUTE	3 =
CUR.LOCATION	Permanent attribute	Integer	PREAMBLE	2
			AMBULANCE.RUN	4

			DISPATCHER	2
			FIND.HOSP	2
			GET.AMB	1
			GET.EMS	1 =
			RECALL	1 =
			TR.ON.SCENE	1
			TRAVEL	1 =
D.BLEEDING.RATE	Temporary attribute	Double	BLEED	2 =
			UPDATE	1 =
D.BLOOD.VOLUME	Temporary attribute	Double	BLEED	5 =
			UPDATE	1 =
D.HCT	Temporary attribute	Double	BLEED	3 =
			UPDATE	1 =
D.O2.DELIVERY	Temporary attribute	Double	BLEED	2 =
			UPDATE	1 =
D.RBC.MASS	Temporary attribute	Double	BLEED	3 =
			UPDATE	1 =
D.SBP	Temporary attribute	Double	BLEED	4 =
			UPDATE	1 =
DEAD	Define to mean		PREAMBLE	1
			BLEED	1
			PASS.TIME	1
			PATIENT	2
DEBUG	Define to mean		PREAMBLE	1
			DISPATCHER	3
			FIND.HOSP	2
			PASS.TIME	1
			PATIENT	2
			RECALL	1
DEF.BP	Temporary attribute	Double	PREAMBLE	2
			PATIENT	1 =
			PT.REPORT	1
DEST.LEVEL	Temporary attribute	Integer	PREAMBLE	2
			AMBULANCE.RUN	3 =
			ASSIGN.AMB	2 =
			FIND.HOSP	2
			GET.PATIENT	1 =
DESTINATION	Temporary attribute	Integer	PREAMBLE	2
			AMBULANCE.RUN	3
			ASSIGN.AMB	1 =
			DISPATCHER	1 =
			FIND.LOC	3
DIM.F	Library routine		LIN.INT.F	1
			NSP.F	1
DISPATCHER	Routine		ACCIDENT	2
			AMBULANCE.RUN	5
			CHECK.ACCIDENT	2
			DISPATCHER	1
			GET.PATIENT	1

DIVERT.COUNTER	Global variable	Integer	PREAMBLE MAIN FIND.HOSP	2 1 1 =
DONE	Routine	Integer	PREAMBLE DONE PASS.TIME PATIENT	1 1 1 1
DONE.RESUS	Define to mean		PREAMBLE PASS.TIME	1 1
EDGE.SET	Set		PREAMBLE BEST.ROUTE GET.NET PRINT.NET	2 2 2 1
EN.ROUTE	Define to mean		PREAMBLE AMBULANCE.RUN PASS.TIME	1 1 2
ERR.F	Library routine		MYGAMMA.F	2
EST.DEATHS	Global variable	Double	PREAMBLE FINAL.REPORT PATIENT	1 1 1 =
EXP.C	Permanent attribute	Double	MYGAMMA.F	1
EXP.F	Library routine		BLEED MYGAMMA.F PATIENT	1 2 2
EXPONENTIAL.F	Library routine		ACCIDENT AMBULANCE.RUN CHOKE.F PATIENT	2 2 1 1
F.AMB.PATIENT.SET	Temporary attribute	Pointer	AMBULANCE.RUN	1
F.H.READY.SET	Permanent attribute	Integer2	GET.AMB	1
FALSE	Define to mean		PREAMBLE AMBULANCE.RUN BUILD.HOSP.LIST CHECK.RED CLEAR.REDS DISPATCHER DONE FIND.HOSP GET.AMB GET.HOSP GET.LIST GO.GREEN GO.OFF.RED GO.RED INIT.ACCIDENT LIVING LEFT NO.PTS PASS.TIME PATIENT	1 3 3 1 2 2 2 2 1 3 1 1 2 1 2 1 2 4
FINAL.REPORT	Routine		MAIN	1

			FINAL.REPORT	1
FIND.HOSP	Routine	Integer	PREAMBLE	1
			DISPATCHER	1
			FIND.HOSP	1
			GET.PATIENT	1
FIND.LOC	Routine	Integer	PREAMBLE	1
			FIND.LOC	1
			RECALL	1
FLIGHT.TIME	Permanent attribute	Double	PREAMBLE	2
			GET.NET	2 =
			GET.TRAVEL.TIME	1
			PRINT.NET	1
FOREVER	Define to mean		PREAMBLE	1
			AMBULANCE.RUN	1
			BEST.ROUTE	1
			BUILD.CALL.LIST	1
			BUILD.HOSP.LIST	1
			GENERATOR	1
FREE	Define to mean		PREAMBLE	1
			BUILD.ROUTE	1
FTIME.F	Routine	Double	PREAMBLE	1
			FTIME.F	1
			GET.NET	1
FULL	Permanent attribute	Integer	PREAMBLE	2
			GET.HOSP	1 =
			LEFT NO.PTS	3 =
G.ACCUM.CAP	Permanent attribute	Double	PREAMBLE	2
			MAIN	1 =
			FINAL.REPORT	1
G.ACCUM.PTS	Permanent attribute	Double	PREAMBLE	2
			MAIN	1 =
			FINAL.REPORT	1
G.MAX.ACCS	Global variable	Double	PREAMBLE	1
			FINAL.REPORT	1
G.MAX.DUR	Global variable	Double	PREAMBLE	1
			FINAL.REPORT	1
G.MAX.MAXPEND	Global variable	Double	PREAMBLE	1
			FINAL.REPORT	1
G.MAX.PTS	Global variable	Double	PREAMBLE	1
			FINAL.REPORT	1
G.MEAN.ACCS	Routine	Double	PREAMBLE	1
			FINAL.REPORT	1
G.MEAN.AMB	Routine	Double	PREAMBLE	1
			FINAL.REPORT	1
G.MEAN.DUR	Routine	Double	PREAMBLE	1
			FINAL.REPORT	1

G.MEAN.MAXPEND	Routine	Double	PREAMBLE FINAL.REPORT	1 1
G.MEAN.PEND	Routine	Double	PREAMBLE FINAL.REPORT	1 1
G.MEAN.PTS	Routine	Double	PREAMBLE FINAL.REPORT	1 1
G.SSQ.CAP	Permanent attribute	Double	PREAMBLE MAIN FINAL.REPORT	2 1 = 1
G.SSQ.PTS	Permanent attribute	Double	PREAMBLE MAIN FINAL.REPORT	2 1 = 1
G.VAR.ACCS	Routine	Double	PREAMBLE FINAL.REPORT	1 1
G.VAR.AMB	Routine	Double	PREAMBLE FINAL.REPORT	1 1
G.VAR.DUR	Routine	Double	PREAMBLE FINAL.REPORT	1 1
G.VAR.MAXPEND	Routine	Double	PREAMBLE	1
G.VAR.PEND	Routine	Double	PREAMBLE FINAL.REPORT	1 1
G.VAR.PTS	Routine	Double	PREAMBLE FINAL.REPORT	1 1
GENERATOR	Process notice		PREAMBLE MAIN GENERATOR	1 1 1
	+ Global variable	Pointer	MAIN	1 =
GET.ACCS	Routine	Pointer	PREAMBLE GET.ACCS INITIALIZE	1 1 1
GET.AMB	Routine	Integer	PREAMBLE DISPATCHER GET.AMB	1 5 1
GET.EMS	Routine		GET.EMS INITIALIZE	1 1
GET.HOSP	Routine		GET.HOSP INITIALIZE	1 1
GET.IV.RATE	Routine	Double	PREAMBLE AMBULANCE.RUN GET.IV.RATE GET.PATIENT	1 3 1 1
GET.LEVEL	Routine	Integer	PREAMBLE AMBULANCE.RUN GET.LEVEL GET.PATIENT	1 1 1 1

GET.LIST	Routine		GET.LIST INITIALIZE	1 2
GET.NET	Routine		GET.NET INITIALIZE	1 1
GET.NUM.AMB	Routine	Integer	PREAMBLE DISPATCHER GET.NUM.AMB	1 3 1
GET.PATIENT	Routine		AMBULANCE.RUN GET.PATIENT	2 1
GET.SIM	Routine		GET.SIM INITIALIZE	1 1
GET.TABLE	Routine	Pointer	PREAMBLE GET.TABLE INITIALIZE	1 1 1
GET.TRAVEL.TIME	Routine	Double	PREAMBLE DISPATCHER GET.AMB GET.TRAVEL.TIME	1 2 1 1
GO.GREEN	Routine	Integer	PREAMBLE CHECK.RED GO.GREEN GO.OFF.RED	1 1 1 1
GO.ITEM	Temporary entity		PREAMBLE BUILD.HOSP.LIST READ.HOSP.LIST	1 2 1
	+ Global variable	Pointer	BUILD.HOSP.LIST READ.HOSP.LIST	10 = 4 =
GO.OFF.RED	Event notice		PREAMBLE CHECK.RED CLEAR.REDS GO.OFF.RED	1 3 1 3
	+ Global variable	Pointer	GO.OFF.RED	1
GO.RED	Routine	Integer	PREAMBLE CHECK.RED GO.RED	1 1 1
GOR.HSP	Temporary attribute	Integer	PREAMBLE CHECK.RED GO.OFF.RED	2 1 = 1 =
GOT.HELO	Temporary attribute	Integer	PREAMBLE DISPATCHER PT.REPORT	2 2 = 1
GR.MAX.PTS	Permanent attribute	Integer	PREAMBLE MAIN FINAL.REPORT	2 1 = 1
GREEN.LIMIT	Global variable	Integer	PREAMBLE GET.SIM GO.GREEN	1 1 = 1
GREEN.SET	Set		PREAMBLE	3

			CHECK.RED	3
			CLEAR.REDS	1
			FIND.HOSP	1
			GET.HOSP	1
			GO.OFF.RED	1
GROUND	Define to mean		PREAMBLE	1
			BUILD.CALL.LIST	2
			DISPATCHER	5
			GET.AMB	2
			GET.EMS	1
			GET.TRAVEL.TIME	1
			PRINT.NET	1
			RECALL	1
H.CTS	Temporary attribute	Integer	PREAMBLE	2
			CTS.F	6 =
			INIT.PT	1 =
H.READY.SET	Set		PREAMBLE	4
			BUILD.CALL.LIST	1
			DISPATCHER	1
			GET.AMB	2
			GET.EMS	1
H.WAITING.SET	Set		PREAMBLE	2
			AMBULANCE.RUN	1
			CHECK.ACCIDENT	1
			DISPATCHER	1
HCT	Temporary attribute	Double	PREAMBLE	2
			BLEED	2
			PATIENT	3 =
			UPDATE	1 =
HCT.0	Define to mean		PREAMBLE	2
			PATIENT	1
HELO.COMING	Temporary attribute	Integer	PREAMBLE	2
			AMBULANCE.RUN	5 =
			DISPATCHER	1 =
HOSP	Temporary attribute	Integer	PREAMBLE	3
			AMBULANCE.RUN	2 =
			DISPATCHER	1 =
			PATIENT	4
			PT.REPORT	1
			PVT.TRAVEL	1 =
			TR.DELIVER.PT	1
			TR.ENROUTE.HOSP	1
			TR.PVT.TR.PT	1
HOSP.BASE	Permanent attribute	Integer	PREAMBLE	2
			DISPATCHER	1
			GET.HOSP	4 =
			GET.PATIENT	1
			PRINT.NET	2
			PVT.TRAVEL	2
HOSP.BASE.SET	Set		PREAMBLE	2
			BUILD.HOSP.LIST	1
			GET.HOSP	2

HOSP.BP	Temporary attribute	Double	PREAMBLE PATIENT PT.REPORT	2 1 = 1
HOSP.ID	Permanent attribute	Integer	PREAMBLE GET.HOSP	2 1 =
HOSP.LEVEL	Temporary attribute	Integer	PREAMBLE BUILD.HOSP.LIST FIND.HOSP READ.HOSP.LIST WRITE.HOSP.LIST	2 2 = 2 1 = 1
HOSP.LIST	Set		PREAMBLE BUILD.HOSP.LIST FIND.HOSP PRINT.NET PVT.TRAVEL READ.HOSP.LIST WRITE.HOSP.LIST	2 4 2 1 2 1 1
HOSP.NAME	Permanent attribute	Text	PREAMBLE FINAL.REPORT FIND.HOSP GET.HOSP PRINT.NET RUN.REPORT TR.DELIVER.PT TR.ENROUTE.HOSP TR.GO.GREEN TR.GO.RED TR.PVT.TR.PT TR.RESUS.PT	2 1 1 1 = 2 1 1 1 1 1 1 1
HOSP.SET	Set		PREAMBLE BUILD.HOSP.LIST GET.HOSP	2 1 1
HOSP.VOLUME	Permanent attribute	Double	PREAMBLE FIND.HOSP GET.HOSP PVT.TRAVEL	2 2 1 = 2
HOSPITAL	Permanent entity + Global variable	Integer2	PREAMBLE GET.HOSP MAIN FINAL.REPORT GET.HOSP RUN.REPORT	1 1 15 = 2 = 10 = 8 =
HOSPITAL.ID	Temporary attribute	Integer	PREAMBLE BUILD.HOSP.LIST FIND.HOSP PRINT.NET PVT.TRAVEL READ.HOSP.LIST WRITE.HOSP.LIST	2 2 = 6 2 3 1 = 1
HOURL.F	Library routine		DISPATCHER FIND.HOSP GENERATOR PASS.TIME PATIENT	3 2 1 1 2

			RECALL	1
			TR.CHECK.IN.ACC	1
			TR.CHECK.IN.AMB	1
			TR.CHECK.OUT.ACC	1
			TR.CHECK.OUT.PT	1
			TR.DELIVER.PT	1
			TR.ENROUTE.HOSP	1
			TR.GO.GREEN	1
			TR.GO.RED	1
			TR.ON.SCENE	1
			TR.PICKUP.PT	2
			TR.PVT.TR.PT	1
			TR.RESUS.PT	1
			TR.SEND.AMB	1
			TR.TO.HOME	1
			TR.UNSTACK.ACC	1
HOURS.V	Permanent attribute	Double	BLEED	1
			CHECK.RED	1
			DONE	1
			GENERATOR	1
			GET.IV.RATE	1
			GET.PATIENT	1
			GO.OFF.RED	1
			INIT.PT	1
			PASS.TIME	2
			PATIENT	6
			PT.REPORT	1
			PVT.TRAVEL	1
			RUN.REPORT	2
HTOL	Global variable	Double	PREAMBLE	1
			BUILD.HOSP.LIST	2
			GET.SIM	1 =
			PRINT.NET	1
IN.USE	Define to mean		PREAMBLE	1
			BUILD.ROUTE	1
INFO.LAG	Global variable	Double	PREAMBLE	1
			ACCIDENT	1
			GET.SIM	1 =
INIT.ACCIDENT	Routine		GENERATOR	1
			INIT.ACCIDENT	1
INIT.PT	Routine		INIT.ACCIDENT	1
			INIT.PT	1
INITIALIZE	Routine		MAIN	1
			INITIALIZE	1
INJURY.LOC	Temporary attribute	Integer	PREAMBLE	2
			INIT.PT	1 =
			PT.REPORT	1
INJURY.TIME	Temporary attribute	Double	PREAMBLE	2
			DONE	1
			PASS.TIME	2
			PATIENT	2 =
			PT.REPORT	1
INT.F	Library routine		DISPATCHER	1

			GET.NUM.AMB	1
			INIT.PT	2
INTEGRATOR.V	Permanent attribute	Integer	GET.SIM	1 =
IS.TO.IV	Define to mean		PREAMBLE	1
			BLEED	1
ISS	Temporary attribute	Integer	PREAMBLE	2
			INIT.PT	9 =
			PATIENT	2
			PT.REPORT	1
IV.RATE	Temporary attribute	Double	PREAMBLE	2
			AMBULANCE.RUN	3 =
			BLEED	1
			GET.PATIENT	1 =
			PATIENT	1
IV.START.TIME	Temporary attribute	Double	PREAMBLE	3
			BLEED	2
			GET.PATIENT	1 =
KIND	Temporary attribute	Integer	PREAMBLE	3
			ACCIDENT	1
			AMBULANCE.RUN	6
			ASSIGN.AMB	1 =
			DISPATCHER	2
			INIT.ACCIDENT	4 =
			TR.CHECK.IN.ACC	1
			TR.CHECK.OUT.ACC	1
			TR.ENROUTE.HOSP	1
			TR.ON.SCENE	1
			TR.PICKUP.PT	1
			TR.SEND.AMB	1
			TR.STACK.ACC	1
			TR.UNSTACK.ACC	1
LAPLACE.F	Routine	Double	PREAMBLE	1
			LAPLACE.F	1
			MYGAMMA.F	1
LAT	Permanent attribute	Double	PREAMBLE	2
			FTIME.F	2
			GET.NET	1 =
LEVEL	Permanent attribute	Integer	PREAMBLE	2
			BUILD.HOSP.LIST	5
			GET.HOSP	1 =
			PRINT.NET	2
LEVEL1	Define to mean		PREAMBLE	1
			GET.LEVEL	1
			GET.PATIENT	1
			PRINT.NET	2
LEVEL2	Define to mean		PREAMBLE	1
			ASSIGN.AMB	1
			PRINT.NET	2
LEVEL2A	Define to mean		PREAMBLE	1
			BUILD.HOSP.LIST	1
			GET.LEVEL	1

			PRINT.NET	2
LEVEL3	Define to mean		PREAMBLE	1
			AMBULANCE.RUN	2
			ASSIGN.AMB	1
			GET.LEVEL	1
			GET.PATIENT	1
			PRINT.NET	2
LIN.INT.F	Routine	Double	PREAMBLE	1
			LIN.INT.F	1
			NSP.F	1
LINES.V	Temporary attribute	Integer	MAIN	4 =
LIVING	Routine	Integer	PREAMBLE	1
			DONE	1
			LIVING	1
			PASS.TIME	1
			PATIENT	2
LL.TIME	Global variable	Double	PREAMBLE	1
			FTIME.F	1
			GET.NET	1
			GET.SIM	1 =
LOG.E.F	Library routine		LAPLACE.F	2
			MYGAMMA.F	4
			NSP.F	1
LOG.NORMAL.F	Library routine		AMBULANCE.RUN	2
			GET.IV.RATE	1
LONG	Permanent attribute	Double	PREAMBLE	2
			FTIME.F	1
			GET.NET	1 =
M.DELIVER	Global variable	Double	PREAMBLE	1
			AMBULANCE.RUN	1
			GET.SIM	1 =
M.PATIENT.COUNTER	Global variable	Integer	PREAMBLE	2
			MAIN	1
			INIT.ACCIDENT	1 =
M.SECURE	Global variable	Double	PREAMBLE	1
			AMBULANCE.RUN	1
			GET.SIM	1 =
MAJOR.CUTOFF	Global variable	Double	PREAMBLE	1
			GET.LEVEL	1
			GET.SIM	2 =
			PRINT.NET	1
MAJOR.TIME	Global variable	Integer	PREAMBLE	1
			GET.PATIENT	1
			GET.SIM	1 =
MAX.F	Library routine		MAIN	1
			GET.IV.RATE	2
			GET.NUM.AMB	1
			INIT.PT	1
			LEFT NO.PTS	1

MAX.RED	Global variable	Integer	PREAMBLE GET.SIM GO.OFF.RED	1 1 = 1
MAX.RED.HOSP	Global variable	Integer	PREAMBLE GET.SIM GO.RED	1 1 = 1
MAX.STEP.V	Permanent attribute	Real	GET.SIM	1 =
MAXRTS	Define to mean		PREAMBLE AMBULANCE.RUN GET.SIM	1 1 2
MDT	Set		PREAMBLE BEST.ROUTE	2 4
MED.COUNTER	Global variable	Integer	PREAMBLE MAIN INIT.ACCIDENT	2 1 1 =
MEDICAL	Define to mean		PREAMBLE ACCIDENT AMBULANCE.RUN DISPATCHER INIT.ACCIDENT	1 1 3 2 2
MIN.AMB	Global variable	Integer	PREAMBLE BUILD.CALL.LIST GET.SIM	1 1 1 =
MIN.F	Library routine		AMBULANCE.RUN GET.PATIENT	1 1
MIN.LENGTH	Global variable	Double	PREAMBLE FINAL.REPORT GENERATOR GET.SIM	1 1 1 1 =
MIN.RED	Global variable	Integer	PREAMBLE CHECK.RED GET.SIM GO.OFF.RED	1 2 1 = 1
MIN.STEP.V	Permanent attribute	Real	GET.SIM	1 =
MINOR.CUTOFF	Global variable	Double	PREAMBLE AMBULANCE.RUN GET.LEVEL GET.SIM PRINT.NET	1 1 1 2 = 1
MINOR.TIME	Global variable	Integer	PREAMBLE GET.PATIENT GET.SIM	1 1 1 =
MINUTE.F	Library routine		DISPATCHER FIND.HOSP GENERATOR PASS.TIME PATIENT RECALL TR.CHECK.IN.ACC	3 2 1 1 2 1 1

			TR.CHECK.IN.AMB	1
			TR.CHECK.OUT.ACC	1
			TR.CHECK.OUT.PT	1
			TR.DELIVER.PT	1
			TR.ENROUTE.HOSP	1
			TR.GO.GREEN	1
			TR.GO.RED	1
			TR.ON.SCENE	1
			TR.PICKUP.PT	2
			TR.PVT.TR.PT	1
			TR.RESUS.PT	1
			TR.SEND.AMB	1
			TR.TO.HOME	1
			TR.UNSTACK.ACC	1
MINUTES.V	Permanent attribute	Double	BLEED	1
			DONE	1
			FTIME.F	1
			GET.IV.RATE	1
			GET.NET	1
			GET.PATIENT	1
			INIT.PT	1
			PASS.TIME	2
			PATIENT	6
			PT.REPORT	1
			PVT.TRAVEL	1
			RUN.REPORT	1
MOD.F	Library routine		NSP.F	1
MYBETA.F	Routine	Double	PREAMBLE	1
			INIT.PT	1
			MYBETA.F	1
MYGAMMA.F	Routine	Double	PREAMBLE	1
			AMBULANCE.RUN	1
			GET.TRAVEL.TIME	2
			MYBETA.F	2
			MYGAMMA.F	1
			PATIENT	1
			PVT.TRAVEL	1
N.ACC.PATIENT.SET	Temporary attribute	Integer2	DISPATCHER	2
N.AMB.PATIENT.SET	Temporary attribute	Integer2	TR.ENROUTE.HOSP	1
N.AMB.SET	Temporary attribute	Integer2	CHECK.ACCIDENT	1
			DISPATCHER	1
N.AMBULANCE	Global variable	Integer2	MAIN	1
			GENERATOR	1
N.CTS	Temporary attribute	Integer	PREAMBLE	2
			CTS.F	1
			INIT.PT	5 =
N.EDGE.SET	Permanent attribute	Integer2	GET.NET	1
N.H.READY.SET	Permanent attribute	Integer2	PREAMBLE	1
			MAIN	1
			GENERATOR	1
N.NODE.AMB.SET	Permanent attribute	Integer2	GET.AMB	1

N.PENDING.SET	Permanent attribute	Integer2	PREAMBLE MAIN DISPATCHER	1 1 1
N.READY.SET	Permanent attribute	Integer2	PREAMBLE MAIN GENERATOR	1 1 1
N.RED.SET	Permanent attribute	Integer2	GO.RED	1
N.TEMP.SET	Permanent attribute	Integer2	BUILD.CALL.LIST BUILD.HOSP.LIST	1 1
NEED.HELO	Temporary attribute	Integer	PREAMBLE GET.PATIENT PT.REPORT	2 1 = 1
NEEDED	Temporary attribute	Integer	PREAMBLE DISPATCHER	2 12 =
NEW.INFO	Define to mean		PREAMBLE ACCIDENT DISPATCHER	1 1 1
NMPD	Define to mean		PREAMBLE FTIME.F	1 2
NO.BLUNT	Global variable	Integer	PREAMBLE MAIN FINAL.REPORT INIT.ACCIDENT	2 1 1 1 =
NO.DEATHS	Global variable	Integer	PREAMBLE MAIN FINAL.REPORT PASS.TIME PATIENT	2 1 1 1 = 2 =
NO.PENED	Global variable	Integer	PREAMBLE MAIN GET.AMB	2 1 1 =
NO.PTS	Permanent attribute	Integer	PREAMBLE GO.OFF.RED LEFT NO.PTS LEFT NO.PTS PATIENT	2 1 1 2 = 2 =
NO.PTS.LEFT	Define to mean		PREAMBLE CHECK.ACCIDENT DISPATCHER	1 1 1
NO.RUNS	Global variable	Integer	PREAMBLE MAIN FINAL.REPORT GET.SIM	1 1 3 1 =
NO.VICTIMS	Temporary attribute	Integer	PREAMBLE DISPATCHER INIT.ACCIDENT	2 1 3 =
NODE	Permanent entity		PREAMBLE BUILD.CALL.LIST	2 5

			BUILD.HOSP.LIST	2
			GET.NET	3
	+ Global variable	Integer2	BEST.ROUTE	2 =
			BUILD.CALL.LIST	8 =
			BUILD.HOSP.LIST	4 =
			GET.NET	2 =
			INIT.ACCIDENT	2 =
			PRINT.NET	9 =
			WRITE.CALL.LIST	3 =
			WRITE.HOSP.LIST	3 =
	+ Implied subscript		GET.AMB	1
NODE.AMB.SET	Set		PREAMBLE	2
			BUILD.CALL.LIST	2
			GET.EMS	1
			TRAVEL	2
NODE.ID	Permanent attribute	Integer	PREAMBLE	2
			GET.NET	1 =
NODE.NAME	Permanent attribute	Text	PREAMBLE	2
			GET.NET	1 =
			PRINT.NET	7
			TR.CHECK.IN.ACC	1
			TR.CHECK.IN.AMB	1
			TR.ON.SCENE	1
NODE.SET	Set		PREAMBLE	2
			BEST.ROUTE	5
NOT.WORKING	Define to mean		PREAMBLE	1
			RECALL	1
NSP.F	Routine	Double	PREAMBLE	1
			GENERATOR	2
			NSP.F	1
NSP.LAST.TIME	Global variable	Double	PREAMBLE	1
			MAIN	1 =
			NSP.F	2 =
NSP.TPRIME	Global variable	Double	PREAMBLE	1
			MAIN	1 =
			NSP.F	3 =
NTRPT	Define to mean		PREAMBLE	1
NULL	Define to mean		PREAMBLE	1
			ACCIDENT	2
			AMBULANCE.RUN	2
			CHECK.ACCIDENT	1
			CHECK.RED	1
			CLEAR.REDS	2
			FIND.HOSP	2
			GET.AMB	2
			GO.OFF.RED	1
NUM.HELO	Global variable	Integer	PREAMBLE	1
			GET.EMS	1 =
			RUN.REPORT	2
O2.DELIVERY	Temporary attribute	Double	PREAMBLE	2
			LIVING	1

			PATIENT UPDATE	1 = 1 =
02.SAT	Temporary attribute	Double	PREAMBLE BLEED INIT.PT PATIENT RESP.SUPPORT	2 1 1 = 1 1 =
02.SAT.0	Define to mean		PREAMBLE	2
ON.SCENE.SET	Set		PREAMBLE CHECK.ACCIDENT DISPATCHER	3 1 1
OPS	Define to mean		PREAMBLE BLEED	1 1
OUT.OF.SERVICE.SET	Set		PREAMBLE	2
OVERRIDE.COUNTER	Global variable	Integer	PREAMBLE MAIN FIND.HOSP	2 1 1 =
P.SECURE	Global variable	Double	PREAMBLE AMBULANCE.RUN GET.SIM	1 1 1 =
PASS.TIME	Routine		PASS.TIME PATIENT	1 4
PATIENT	Process notice		PREAMBLE INIT.ACCIDENT PATIENT BLEED DONE PASS.TIME PATIENT	1 2 1 1 1 1 2
	+ Implied subscript		PATIENT	1
	+ Global variable	Pointer		
PATIENT.COUNTER	Global variable	Integer	PREAMBLE INIT.ACCIDENT	1 3 =
PATIENT.SET	Set		PREAMBLE GENERATOR INIT.ACCIDENT PATIENT	3 1 1 2
PENDED	Temporary attribute	Integer	PREAMBLE GET.AMB INIT.ACCIDENT	2 2 = 1 =
PENDING.SET	Set		PREAMBLE DISPATCHER GET.AMB	4 5 2
PHASE	Temporary attribute	Integer	PREAMBLE AMBULANCE.RUN DONE GET.PATIENT INIT.PT PASS.TIME PATIENT	2 2 = 2 1 = 1 = 7 = 3 =

			PVT.TRAVEL	1 =
POISSON.F	Library routine		INIT.ACCIDENT	1
PRINT.NET	Routine		INITIALIZE	1
			PRINT.NET	1
PRIVATE	Define to mean		PREAMBLE	1
			PVT.TRAVEL	1
PROB.SURV	Temporary attribute	Double	PREAMBLE	2
			PATIENT	4 =
			PT.REPORT	1
PROP.ACCS	Global variable	Double	PREAMBLE	1
			INIT.ACCIDENT	1
			INITIALIZE	1 =
PSTF	Define to mean		PREAMBLE	1
PT.ID	Temporary attribute	Integer	PREAMBLE	2
			BLEED	1
			INIT.ACCIDENT	1 =
			PASS.TIME	1
			PATIENT	2
			PT.REPORT	1
			TR.CHECK.IN.PT	1
			TR.CHECK.OUT.PT	1
			TR.PICKUP.PT	1
			TR.PVT.TR.PT	1
			TR.RESUS.PT	1
PT.REPORT	Routine		PATIENT	2
			PT.REPORT	1
PTS.PER.ACC	Global variable	Double	PREAMBLE	1
			GET.SIM	1 =
PVT.TR.ARRIVE	Temporary attribute	Double	PREAMBLE	2
			DONE	1
			PASS.TIME	1
			PVT.TRAVEL	1 =
PVT.TR.SET	Set		PREAMBLE	3
			PATIENT	1
			PVT.TRAVEL	1
PVT.TRAV	Define to mean		PREAMBLE	1
			DONE	1
			PASS.TIME	2
			PATIENT	1
			PVT.TRAVEL	1
PVT.TRAVEL	Routine		ACCIDENT	1
			PVT.TRAVEL	1
R.ACCUM.CAP	Permanent attribute	Double	PREAMBLE	2
			MAIN	2 =
			LEFT NO.PTS	1 =
			RUN.REPORT	2
R.ACCUM.PTS	Permanent attribute	Double	PREAMBLE	2
			MAIN	2 =

			LEFT NO.PTS	1 =
			RUN.REPORT	2
R.CTS	Temporary attribute	Integer	PREAMBLE	2
			CTS.F	1
			INIT.PT	5 =
			RESP.SUPPORT	2 =
R.EST.DEATHS	Global variable	Double	PREAMBLE	1
			MAIN	1 =
			PATIENT	1 =
			RUN.REPORT	2
R.IDLE.AMB	Routine	Double	PREAMBLE	1
			MAIN	1
			RUN.REPORT	2
R.IDLE.HELO	Routine	Double	PREAMBLE	1
			RUN.REPORT	2
R.MAX.PENDING	Global variable	Double	PREAMBLE	1
			MAIN	1
			RUN.REPORT	1
R.MAX.PTS	Permanent attribute	Integer	PREAMBLE	2
			MAIN	3 =
			LEFT NO.PTS	1 =
			RUN.REPORT	1
R.MEAN.PENDING	Routine	Double	PREAMBLE	1
			MAIN	1
			RUN.REPORT	2
R.MEAN.RED	Routine	Double	PREAMBLE	1
			RUN.REPORT	2
R.NO.ACCS	Global variable	Double	PREAMBLE	1
			MAIN	1
			GENERATOR	1
			RUN.REPORT	2
R.NO.BLUNT	Global variable	Double	PREAMBLE	1
			RUN.REPORT	2
R.NO.DEATHS	Global variable	Double	PREAMBLE	1
			RUN.REPORT	2
R.NO.DIVERT	Global variable	Double	PREAMBLE	1
			RUN.REPORT	2
R.NO.M.PTS	Global variable	Double	PREAMBLE	1
R.NO.MEDS	Global variable	Double	PREAMBLE	1
			RUN.REPORT	1
R.NO.OVERRIDE	Global variable	Double	PREAMBLE	1
			RUN.REPORT	2
R.NO.PENDED	Global variable	Double	PREAMBLE	1
			RUN.REPORT	4
R.NO.T.PTS	Global variable	Double	PREAMBLE	1
			MAIN	1

			RUN.REPORT	2
R.PT	Temporary attribute	Pointer	PREAMBLE	2
			GET.PATIENT	1 =
			PATIENT	1 =
R.SSQ.CAP	Global variable	Double	PREAMBLE	1
R.SSQ.PTS	Global variable	Double	PREAMBLE	1
RANDOM.F	Library routine		ACCIDENT	1
			AMBULANCE.RUN	1
			FIND.HOSP	1
			INIT.ACCIDENT	3
			INIT.PT	2
			LAPLACE.F	1
			MYGAMMA.F	3
			NSP.F	1
			PVT.TRAVEL	1
RANGE	Global variable	Double	PREAMBLE	1
			GET.NET	1 =
RBC.MASS	Temporary attribute	Double	PREAMBLE	2
			BLEED	1
			PATIENT	1 =
			UPDATE	1 =
READ.CALL.LIST	Routine		INITIALIZE	1
			READ.CALL.LIST	1
READ.HOSP.LIST	Routine		INITIALIZE	1
			READ.HOSP.LIST	1
READY.SET	Set		PREAMBLE	3
			DISPATCHER	1
			GET.AMB	3
			GET.EMS	1
RECALL	Routine		DISPATCHER	2
			RECALL	1
RED	Permanent attribute	Integer	PREAMBLE	3
			MAIN	1
			CHECK.RED	2 =
			CLEAR.REDS	1 =
			GET.HOSP	1 =
			GO.OFF.RED	1 =
RED.LIMIT	Global variable	Integer	PREAMBLE	1
			GET.SIM	1 =
			GO.RED	1
RED.SET	Set		PREAMBLE	3
			CHECK.RED	2
			CLEAR.REDS	2
			FIND.HOSP	1
			GENERATOR	1
			GO.OFF.RED	1
RED.START	Permanent attribute	Double	PREAMBLE	2
			CHECK.RED	2 =
			GO.OFF.RED	1

RED.TODAY	Permanent attribute	Integer	PREAMBLE CHECK.RED CLEAR.REDS GET.HOSP GO.RED	2 1 = 1 = 1 = 1
REDS.PER.DAY	Global variable	Integer	PREAMBLE GET.SIM GO.RED	1 1 = 1
REL.ERR.V	Permanent attribute	Real	GET.SIM	1 =
REQ.AMB	Define to mean		PREAMBLE ACCIDENT DISPATCHER	1 1 1
REQ.CAR	Define to mean		PREAMBLE	1
REQ.HELO	Define to mean		PREAMBLE DISPATCHER GET.PATIENT	1 1 1
REQ.HELP	Define to mean		PREAMBLE CHECK.ACCIDENT DISPATCHER	1 1 1
RESCUE.BP	Temporary attribute	Double	PREAMBLE PATIENT PT.REPORT	2 1 = 1
RESP.SUPPORT	Event notice		PREAMBLE GET.PATIENT PATIENT RESP.SUPPORT	1 1 1 1
	+ Global variable	Pointer	GET.PATIENT PATIENT RESP.SUPPORT	1 = 1 = 1
RESP.TIME	Temporary attribute	Double	PREAMBLE GET.PATIENT	2 2 =
RESUS	Define to mean		PREAMBLE AMBULANCE.RUN DONE PASS.TIME PATIENT	1 1 1 4 1
RESUS.PATIENT.SET	Set		PREAMBLE AMBULANCE.RUN PATIENT	3 1 2
RESUS.TIME	Temporary attribute	Double	PREAMBLE DONE PASS.TIME PATIENT PT.REPORT	2 1 1 3 = 1
ROPENERR.V	Temporary attribute	Integer	GET.LIST	1
ROUTE	Set		PREAMBLE BUILD.ROUTE CHOKE.F FIND.LOC	3 2 1 2

RUN	Global variable	Integer	PREAMBLE MAIN GENERATOR PT.REPORT RUN.REPORT	1 1 = 2 1 2
RUN.COUNTER	Global variable	Integer	PREAMBLE ASSIGN.AMB	1 1 =
RUN.ID	Temporary attribute	Integer	PREAMBLE	2
RUN.REPORT	Routine		MAIN RUN.REPORT	1 1
RUNGE.KUTTA.R	Library routine		GET.SIM	1
S.DELIVER	Global variable	Double	PREAMBLE AMBULANCE.RUN GET.SIM	1 1 1 =
S.MAX.PTS	Permanent attribute	Integer	PREAMBLE MAIN FINAL.REPORT	2 1 = 1
S.SECURE	Global variable	Double	PREAMBLE AMBULANCE.RUN GET.SIM	1 1 1 =
SBP	Temporary attribute	Double	PREAMBLE BLEED CTS.F GET.IV.RATE PATIENT UPDATE	2 2 4 2 5 = 1 =
SBP.0	Define to mean		PREAMBLE BLEED GET.IV.RATE PATIENT	2 2 2 1
SCENE.RX	Define to mean		PREAMBLE GET.PATIENT PASS.TIME	1 1 1
SCENE.TIME	Temporary attribute	Double	PREAMBLE AMBULANCE.RUN DONE GET.PATIENT PASS.TIME PATIENT PT.REPORT	3 6 = 1 2 1 1 = 1
SEED.V	Permanent attribute	Integer	GET.SIM	3 =
SENT	Temporary attribute	Integer	PREAMBLE DISPATCHER GET.AMB	2 5 1 =
SIGN.F	Library routine		BLEED	1
SINK	Temporary attribute	Integer	PREAMBLE BEST.ROUTE BUILD.ROUTE	2 6 1 =

			GET.NET	1 =
			PRINT.NET	1
SITE	Temporary attribute	Integer	PREAMBLE	2
			ASSIGN.AMB	1
			GET.AMB	1
			GET.PATIENT	1
			INIT.ACCIDENT	1 =
			INIT.PT	1
			PVT.TRAVEL	4
			TR.CHECK.IN.ACC	1
SOURCE	Temporary attribute	Integer	PREAMBLE	2
			BEST.ROUTE	2
			BUILD.ROUTE	1 =
			GET.NET	2 =
SQRT.F	Library routine		ADJ.TIME.F	1
			FINAL.REPORT	3
			FTIME.F	1
			INIT.PT	1
			MYGAMMA.F	1
SRC	Temporary attribute	Integer	PREAMBLE	2
			AMBULANCE.RUN	1 =
			FIND.LOC	3
SSPND	Define to mean		PREAMBLE	1
STA.A	Temporary attribute	Integer2	AMBULANCE.RUN	1
START.TIME	Global variable	Double	PREAMBLE	1
			MAIN	2 =
			GENERATOR	1
STATUS	Temporary attribute	Integer	PREAMBLE	2
			AMBULANCE.RUN	2 =
			RECALL	1 =
T.ON.SCENE	Global variable	Double	PREAMBLE	1
			AMBULANCE.RUN	2
			GET.SIM	1 =
T.PATIENT.COUNTER	Global variable	Integer	PREAMBLE	2
			MAIN	1
			INIT.ACCIDENT	1 =
T.RESUS	Global variable	Double	PREAMBLE	1
			GET.SIM	1 =
			PATIENT	2
T.TO.PT	Global variable	Double	PREAMBLE	1
			AMBULANCE.RUN	1
			GET.SIM	1 =
T.TX	Global variable	Double	PREAMBLE	1
			GET.SIM	1 =
			PATIENT	1
TCONST	Define to mean		PREAMBLE	1
			BLEED	1
TCRUISE	Global variable	Double	PREAMBLE	1

			ADJ.TIME.F	3
			GET.SIM	1 =
TEMP.SET	Set		PREAMBLE	2
			BUILD.CALL.LIST	7
			BUILD.HOSP.LIST	5
TIME.A	Temporary attribute	Double	AMBULANCE.RUN	3 =
			DISPATCHER	3 =
			FIND.LOC	1
			GENERATOR	1
			RECALL	2 =
TIME.V	Permanent attribute	Double	MAIN	3 =
			ACCIDENT	1
			AMBULANCE.RUN	1
			BLEED	2
			CHECK.RED	2
			DISPATCHER	6
			DONE	2
			FIND.HOSP	2
			GENERATOR	3
			GET.PATIENT	1
			GO.OFF.RED	1
			INIT.ACCIDENT	1
			LEFT.NO.PTS	2
			PASS.TIME	4
			PATIENT	14
			PVT.TRAVEL	1
			RECALL	2
			RUN.REPORT	1
			TR.CHECK.IN.ACC	1
			TR.CHECK.IN.AMB	1
			TR.CHECK.OUT.ACC	1
			TR.CHECK.OUT.PT	1
			TR.DELIVER.PT	1
			TR.ENROUTE.HOSP	1
			TR.GO.GREEN	1
			TR.GO.RED	1
			TR.ON.SCENE	1
			TR.PICKUP.PT	2
			TR.PVT.TR.PT	1
			TR.RESUS.PT	1
			TR.SEND.AMB	1
			TR.TO.HOME	1
			TR.UNSTACK.ACC	1
TMP.ACCS	Global variable	Integer	PREAMBLE	2
			MAIN	1 =
TMP.AMBS	Global variable	Double	PREAMBLE	2
			MAIN	1 =
TMP.DUR	Global variable	Double	PREAMBLE	2
			MAIN	3 =
			RUN.REPORT	6
TMP.MN.PEND	Global variable	Double	PREAMBLE	2
			MAIN	1 =
TMP.MX.PEND	Global variable	Integer	PREAMBLE	2
			MAIN	1 =

TMP.PTS	Global variable	Integer	PREAMBLE MAIN	2 1 =
TO.BASE	Define to mean		PREAMBLE AMBULANCE.RUN DISPATCHER	1 1 1
TO.BASE.SET	Set		PREAMBLE DISPATCHER	3 2
TO.HOSP	Define to mean		PREAMBLE AMBULANCE.RUN DISPATCHER	1 1 1
TO.HOSP.SET	Set		PREAMBLE DISPATCHER	3 2
TO.SCENE	Define to mean		PREAMBLE	1
TO.SCENE.SET	Set		PREAMBLE ASSIGN.AMB DISPATCHER RECALL	3 1 1 2
TOD	Temporary attribute	Double	PREAMBLE PASS.TIME PATIENT PT.REPORT	2 1 = 1 = 1
TOT.COUNTER	Global variable	Integer	PREAMBLE GENERATOR	1 2 =
TR.CHECK.IN.ACC	Routine		PREAMBLE INIT.ACCIDENT TR.CHECK.IN.ACC	1 1 1
TR.CHECK.IN.AMB	Routine		PREAMBLE DISPATCHER GET.EMS TR.CHECK.IN.AMB	2 2 2 1
TR.CHECK.IN.PT	Routine		PREAMBLE INIT.ACCIDENT TR.CHECK.IN.PT	1 1 1
TR.CHECK.OUT.ACC	Routine		PREAMBLE ACCIDENT TR.CHECK.OUT.ACC	1 1 1
TR.CHECK.OUT.PT	Routine		PREAMBLE PATIENT TR.CHECK.OUT.PT	1 2 1
TR.DELIVER.PT	Routine		PREAMBLE DISPATCHER TR.DELIVER.PT	1 1 1
TR.ENROUTE.HOSP	Routine		PREAMBLE DISPATCHER TR.ENROUTE.HOSP	1 1 1
TR.GO.GREEN	Routine		PREAMBLE CHECK.RED	1 1

			CLEAR.REDS	1
			GET.HOSP	1
			GO.OFF.RED	1
			TR.GO.GREEN	1
TR.GO.RED	Routine		PREAMBLE	1
			CHECK.RED	1
			TR.GO.RED	1
TR.ON.SCENE	Routine		PREAMBLE	1
			DISPATCHER	1
			TR.ON.SCENE	1
TR.PICKUP.PT	Routine		PREAMBLE	1
			AMBULANCE.RUN	1
			GET.PATIENT	1
			TR.PICKUP.PT	1
TR.PROP	Global variable	Double	PREAMBLE	1
			GET.SIM	1 =
			INIT.ACCIDENT	1
TR.PVT.TR.PT	Routine		PREAMBLE	1
			PVT.TRAVEL	1
			TR.PVT.TR.PT	1
TR.RESUS.PT	Routine		PREAMBLE	1
			AMBULANCE.RUN	1
			PATIENT	1
			TR.RESUS.PT	1
TR.SEND.AMB	Routine		PREAMBLE	1
			ASSIGN.AMB	1
			TR.SEND.AMB	1
TR.STACK.ACC	Routine		PREAMBLE	1
			GET.AMB	1
			TR.STACK.ACC	1
TR.TO.HOME	Routine		PREAMBLE	1
			DISPATCHER	1
			TR.TO.HOME	1
TR.UNSTACK.ACC	Routine		PREAMBLE	1
			DISPATCHER	3
			TR.UNSTACK.ACC	1
TRANSIT.TIME	Permanent attribute	Double	PREAMBLE	2
			BUILD.CALL.LIST	3
			BUILD.HOSP.LIST	3
			GET.NET	2 =
			GET.PATIENT	1
			GET.TRAVEL.TIME	1
			PRINT.NET	2
			PVT.TRAVEL	1
TRANSP.MODE	Temporary attribute	Integer	PREAMBLE	2
			AMBULANCE.RUN	1 =
			GET.PATIENT	1 =
			PT.REPORT	1
			PVT.TRAVEL	1 =
TRANSP.TIME	Temporary attribute	Double	PREAMBLE	2

			DONE	1
			PASS.TIME	1
			PATIENT	2 =
			PT.REPORT	1
TRAUMA	Define to mean		PREAMBLE	1
			AMBULANCE.RUN	3
			INIT.ACCIDENT	2
			TR.CHECK.IN.ACC	1
			TR.CHECK.OUT.ACC	1
			TR.ENROUTE.HOSP	1
			TR.ON.SCENE	1
			TR.PICKUP.PT	1
			TR.SEND.AMB	1
			TR.STACK.ACC	1
			TR.UNSTACK.ACC	1
TRAVEL	Routine		AMBULANCE.RUN	3
			TRAVEL	1
TRAVEL.TIME	Permanent attribute	Double	PREAMBLE	2
			DISPATCHER	2 =
			FIND.LOC	1
			GET.AMB	1 =
			TRAVEL	1
TRF.RATE	Temporary attribute	Double	PREAMBLE	2
			PATIENT	1 =
TRF.START.TIME	Temporary attribute	Double	PREAMBLE	2
			PATIENT	1 =
TRUE	Define to mean		PREAMBLE	1
			ACCIDENT	1
			AMBULANCE.RUN	4
			BUILD.HOSP.LIST	5
			CHECK.RED	3
			DISPATCHER	2
			DONE	5
			FIND.HOSP	1
			GET.AMB	1
			GET.PATIENT	2
			GO.GREEN	1
			GO.RED	2
			INIT.ACCIDENT	1
			LIVING	1
			LEFT.NO.PTS	1
			PATIENT	1
			PVT.TRAVEL	1
TRUNC.F	Library routine		NSP.F	1
TX.TIME	Temporary attribute	Double	PREAMBLE	2
			PATIENT	2 =
			PT.REPORT	1
TYPE	Permanent attribute	Integer	PREAMBLE	2
			AMBULANCE.RUN	4
			ASSIGN.AMB	1
			BUILD.CALL.LIST	2
			DISPATCHER	3
			GET.AMB	3
			GET.EMS	3 =

			GET.PATIENT	1
			PRINT.NET	1
			RECALL	1
UIB.R	Implied subscript		GET.LIST	1
UIB.W	Implied subscript		MAIN	4
			PRINT.NET	6
			RUN.REPORT	1
UNIFORM.F	Library routine		DISPATCHER	1
UPDATE	Routine		PASS.TIME	1
			PATIENT	1
			UPDATE	1
UPDATE.TIME	Permanent attribute	Double	PREAMBLE	2
			LEFT NO.PTS	2 =
UPDATED	Temporary attribute	Integer	PREAMBLE	2
			ACCIDENT	1 =
			DISPATCHER	1
V.SET	Set		PREAMBLE	2
			BEST.ROUTE	9
WAIT.TIME	Temporary attribute	Double	PREAMBLE	2
			DONE	1
			PASS.TIME	1
			PATIENT	1 =
			PT.REPORT	1
WAITING	Define to mean		PREAMBLE	1
			INIT.PT	1
WEEKDAY.F	Library routine		GENERATOR	1
			TR.RESUS.PT	1
WEIGHT	Temporary attribute	Double	PREAMBLE	2
			BEST.ROUTE	4
			BUILD.ROUTE	1 =
			FIND.LOC	2
			GET.NET	2 =
			PRINT.NET	1
WORKING	Define to mean		PREAMBLE	1
			AMBULANCE.RUN	2
WRITE.CALL.LIST	Routine		INITIALIZE	1
			WRITE.CALL.LIST	1
WRITE.HOSP.LIST	Routine		INITIALIZE	1
			WRITE.HOSP.LIST	1
WRK	Define to mean		PREAMBLE	1
			AMBULANCE.RUN	1

Appendix 3

Data Files

```
ems.dat -- general simulation control information

4      ;number of runs -- must be at least 2 for valid summary stats
.25    ;min number of days in a run
.5      ;trauma proportion
.5      ;average no. pts / accident
3.0    ;minutes before alarm turned in
3.0    ;minutes til 1st responder calls disp w/ accurate info
6.15   ;6.82 ;mean & sd secure time
.097   ;prob of needing secure time
2.22   ;minutes til pt found
10.4   ;minutes on scene
2.60   ;1.95 ;mean & sd deliver time
15.0   ;resus time
5.0    ;transfer time
5      ;TCRUISE -- time in minutes to attain cruising speed
3      ;min.amb -- minimum no. of ambulance's on a call list
.35    ;atol -- proportional diff in times deemed negligible
.30    ;htol -- same
.95    ;% of cts cutoff for minor trauma--note rts range of 0 to 7.8408
.90    ;cts <= this value goes to level 1 center
39     ;minor.time
19     ;major.time
4.0    ;mean launch+land time for helo
1 1 2  ;red.limit, green.limit, max.red.hosp
1 4 1  ;min and max red time in hours, no.reds per day
8.00   ; time of day to clear red status (assume sim starts at MN)

cont.dat -- continuous simulation control
.0003472 ;max.step ~0.5 minute
.00001157 ;min.step ~1.0 second
.1       ;abs.error
.01      ;rel.error

amb.dat -- ambulances
41       ;no. ambulances
1 4 D1   ;type (1=ground, 2=air), node of base, name
1 1 D4   ;letter = county, # = unit ID
1 1 D9
1 3 D7
1 6 D10
1 2 D13
1 10 D19
1 14 D20
1 19 D22
1 5 D23
1 8 D24
1 18 D25
1 7 D26
1 15 D28
1 11 D30
1 20 D32
1 9 D35
```

```

1 12 D41
1 17 D42
1 16 D50
1 13 D71
1 26 N1
1 26 N2
1 25 N3
1 24 N4
1 23 N5
1 22 N6
1 21 B32
1 29 C70
1 27 C72
1 30 C74
1 28 C76
1 31 P1
1 34 S80
1 34 S83
1 32 S81
1 33 S82
1 36 S88
1 37 G1
2 1 H1
2 2 H2

```

```

hosp.dat -- hospital information
12 ;no. hospitals -- must agree w/ no. of lines following!
UMC 1 95 1 5 1 ;University |name, base, vol in 1000's, level, tcenter
BAP 2 35 2 3 1 ;Baptist |capacity, can.divert (1=yes, 0=no)
MMC 14 30 3 2 1 ;Memorial |watch coding! 4 => level 3!
MTH 1 15 4 1 1 ;Methodist
STL 15 15 3 1 1 ;St Luke's
BCH 13 15 3 1 1 ;Beaches
STV 6 35 3 2 1 ;St Vincent's
RVS 6 12 4 1 1 ;Riverside
HUM 28 18 3 1 0 ;Humana (Orange Park Community)
NGH 26 15 4 1 0 ;Nassau General
FLG 34 18 4 1 0 ;Flagler
PCH 31 18 4 1 0 ;Putnam Community

```

```

net.dat -- nodes and arcs
37 ; n.node -- centered around Rescue stations
1 SPF 30.347 81.639 ;Springfield |node, name, lat, long
2 SBK 30.317 81.623 ;Southbank
3 WJX 30.347 81.665 ;West Jacksonville
4 EJX 30.347 81.623 ;East Jacksonville
5 AVD 30.319 81.672 ;Avondale
6 RVS 30.308 81.663 ;Riverside
7 PKV 30.364 81.718 ;Pickettville
8 LMT 30.415 81.67 ;Lem Turner
9 OWY 30.466 81.619 ;Oceanway
10 ARL 30.338 81.585 ;Arlington
11 RGY 30.323 81.551 ;Regency
12 ATB 30.33 81.408 ;Atlantic Beach
13 JXB 30.289 81.405 ;Jacksonville Beach
14 SSD 30.29 81.579 ;Southside
15 JTB 30.25 81.575 ;J Turner Butler
16 SPL 30.289 81.449 ;San Pablo
17 MND 30.17 81.609 ;Mandarin
18 TMQ 30.25 81.704 ;Timuquana
19 WSD 30.25 81.734 ;Westside
20 MWH 30.321 81.773 ;Marietta-Whitehouse
21 MCL 30.289 82.148 ;McClenny

```

22	BRY	30.398	81.985	;Bryceville
23	CAL	30.56	81.889	;Callahan
24	HIL	30.687	81.966	;Hilliard
25	YUL	30.628	81.569	;Yulee
26	FDB	30.67	81.443	;Fernandina Beach
27	OPE	30.163	81.685	;Orange Park East
28	OPW	30.178	81.7	;Orange Park West
29	GCS	30	81.587	;Green Cove Springs
30	MBG	30.064	81.875	;Middleburg
31	PLK	29.648	81.636	;Palatka
32	PVB	30.238	81.398	;Ponte Vedra Beach
33	SWZ	30.102	81.5	;Switzerland
34	STA	29.867	81.347	;St Augustine
35	CRB	29.83	81.219	;Crescent Beach
36	HST	29.79	81.5	;Hastings
37	KGB	30.79	81.545	;King's Bay

150 600 ;airspeed, range

1	2	10	5.0
1	3	5	0.0
1	4	5	0.0
1	5	10	0.0
1	6	10	0.0
1	8	10	0.0
1	9	10	0.0
2	1	10	5.0
2	4	10	5.0
2	10	10	0.0
2	14	15	0.0
2	15	10	0.0
2	17	20	0.0
2	5	10	5.0
3	1	5	0.0
3	19	20	0.0
3	20	20	0.0
3	7	5	0.0
3	23	20	0.0
3	8	10	0.0
4	1	5	0.0
4	10	5	5.0
4	2	10	5.0
5	1	10	0.0
5	2	10	0.0
5	6	5	0.0
5	18	10	0.0
5	19	10	0.0
5	20	10	0.0
6	1	10	0.0
6	5	5	0.0
7	3	5	0.0
7	19	15	0.0
7	20	15	0.0
7	8	10	0.0
8	1	10	0.0
8	3	10	0.0
8	7	10	0.0
8	23	20	0.0
8	9	10	0.0
9	1	10	0.0
9	8	10	0.0
9	25	20	0.0
9	11	15	0.0
10	4	5	5.0
10	11	5	0.0

10	14	10	0.0		
10	2	10	0.0		
11	10	5	0.0		
11	9	15	0.0		
11	12	20	0.0		
11	15	15	0.0		
11	14	10	0.0		
12	11	20	0.0		
12	16	10	0.0		
12	13	10	0.0		
13	12	10	0.0		
13	16	5	0.0		
13	32	10	0.0		
13	15	15	0.0		
14	2	15	0.0		
14	10	10	0.0		
14	11	10	0.0		
14	16	15	0.0		
14	15	7.5	0.0		
15	2	10	0.0		
15	14	7.5	0.0		
15	11	15	0.0		
15	17	10	0.0		
15	13	15	0.0		
16	14	15	0.0		
16	12	10	0.0		
16	13	5	0.0		
17	2	20	0.0		
17	15	10	0.0		
17	33	20	0.0		
17	27	20	5.0		
18	19	10	0.0		
18	5	10	0.0		
18		27		15	0.0
19	28	15	0.0		
19	20	15	0.0		
19	7	15	0.0		
19	3	20	0.0		
19	5	10	0.0		
19	18	10	0.0		
20	19	15	0.0		
20	5	10	0.0		
20	3	20	0.0		
20	7	15	0.0		
20	22	15	0.0		
20	21	20	0.0		
21	20	20	0.0		
22	20	15	0.0		
22	23	20	0.0		
23	22	20	0.0		
23	24	15	0.0		
23	25	20	0.0		
23	8	20	0.0		
23	3	20	0.0		
24	23	15	0.0		
25	23	20	0.0		
25		9		20	0.0
25		26		20	0.0
25	37	25	0.0		
26		25		20	0.0
27		17		20	5.0
27		18		15	0.0
27		29		15	0.0

27	28	10	0.0
28	19	15	0.0
28	27	10	0.0
28	30	20	0.0
29	27	15	0.0
29	30	20	0.0
29	31	30	0.0
29	33	20	0.0
30	28	20	0.0
30	29	20	0.0
31	29	30	0.0
31	36	20	0.0
32	13	10	0.0
32	34	35	0.0
33	17	20	0.0
33	29	20	0.0
33	34	35	0.0
33	36	30	0.0
34	33	35	0.0
34	32	35	0.0
34	35	20	0.0
34	36	25	0.0
35	34	20	0.0
35	36	20	0.0
36	35	20	0.0
36	33	30	0.0
37	25	25	0.0

space.dat -- relative incidents per node in node order

5
5
5
5
4
2
2
4
2
3
2
2
4
4
2
2
3
3
4
2
2
1
2
1
1
4
4
4
3
2
4
3
1
4

2
1
1

```
wkrate.dat -- time distribution of incidents
84      ;number of entries
0       0      ;first number is time,
2       4.77
4       3.81   ;second is number of events
6       1.14
8       2.07   ;in that time period
10      2.55
12      3.18   ;assumed to "wrap around" after reaching the last
14      3.86
16      4.55   ;time period
18      5.66
20      5.34
22      5.28
24      5.78
26      3.88
28      3.09
30      .93
32      1.68
34      2.07
36      2.59
38      3.14
40      3.69
42      4.60
44      4.34
46      4.29
48      4.70
50      3.88
52      3.09
54      .93
56      1.68
58      2.07
60      2.59
62      3.14
64      3.69
66      4.60
68      4.34
70      4.29
72      4.70
74      4.03
76      3.21
78      .96
80      1.75
82      2.15
84      2.68
86      3.26
88      3.83
90      4.78
92      4.51
94      4.46
96      4.88
98      4.03
100     3.21
102     .96
104     1.75
106     2.15
108     2.68
110     3.26
112     3.83
```


114	4.78
116	4.51
118	4.46
120	4.88
122	4.17
124	3.33
126	1.00
128	1.81
130	2.23
132	2.78
134	3.38
136	3.98
138	4.96
140	4.67
142	4.62
144	5.06
146	5.07
148	4.04
150	1.21
152	2.20
154	2.71
156	3.38
158	4.11
160	4.83
162	6.02
164	5.68
166	5.61
168	6.14

```

seed.dat -- random number stream seeds
28      ;no.streams -- must agree w/ no. lines -- these are 1M apart
683743814
604901985
726466604
622401386
1645973084
1901633463
67784357
2026948561
1545929719
547070247
1110948479
1400311458
1471803249
1232207518
195239450
281826375
416426318
380841429
1055454678
711617330
1416275180
788018608
1357689651
2130853749
152149214
550317865
32645035
871378447

```

The following two files are produced by the best.route routine

```
call.dat -- node, ambulances to be called in order
1 3 2 4 1 40 41
2 6 2 3 1 10 7 14 40 41
3 4 2 3 13 40 41
4 1 2 3 7 40 41
5 10 5 2 3 6 12 9 16 40 41
6 5 10 2 3 40 41
7 13 4 2 3 11 40 41
8 11 2 3 4 13 17 40 41
9 17 2 3 11 40 41
10 7 1 15 40 41
11 15 7 1 8 40 41
12 18 21 20 40 41
13 21 20 18 36 40 41
14 8 14 7 15 40 41
15 14 8 6 19 40 41
16 20 21 18 40 41
17 19 14 8 40 41
18 12 10 9 40 41
19 9 10 12 40 41
20 16 10 5 13 9 27 40 41
21 28 16 10 40 41
22 27 16 26 40 41
23 26 25 4 11 27 24 40 41
24 25 26 4 11 27 24 40 41
25 24 17 26 22 23 40 41
26 23 22 24 40 41
27 30 32 12 29 40 41
28 32 30 9 40 41
29 29 30 31 37 40 41
30 31 32 29 40 41
31 33 38 29 40 41
32 36 21 20 40 41
33 37 19 29 40 41
34 35 34 38 40 41
35 34 35 38 40 41
36 38 37 34 35 40 41
37 39 24 17 26 22 23 40 41
```

```
go.dat -- node, hospital id, hosp level, . . .
1 4 4 1 1 2 2 7 3
2 2 2 1 1 4 4 5 3
3 1 1 4 4 2 2 7 3
4 1 1 4 4 2 2
5 7 3 8 4 1 1
6 8 4 7 3 1 1
7 1 1 4 4 2 2 7 3
8 1 1 4 4 2 2 7 3
9 1 1 4 4 2 2 7 3 3 3
10 1 1 4 4 2 2 3 3
11 3 3 1 1
12 6 3 1 1
13 6 3 1 1
14 3 3 5 3 1 1
15 5 3 3 3 1 1
16 6 3 1 1
17 5 3 1 1
18 7 3 8 4 1 1 4 4 2 2
19 7 3 8 4 9 3 1 1 4 4 2 2
20 7 3 8 4 1 1 4 4 2 2
21 7 3 8 4 1 1 4 4 2 2
```

22	7	3	8	4	1	1	4	4	2	2							
23	1	1	4	4	2	2	7	3	3	3	5	3					
24	1	1	4	4	2	2	7	3	8	4	3	3	5	3			
25	10	4	1	1	2	2	7	3	3	3	5	3					
26	10	4	1	1	2	2	7	3	3	3	5	3					
27	9	3	1	1													
28	9	3	1	1													
29	9	3	12	4	1	1											
30	9	3	1	1													
31	12	4	9	3	1	1											
32	6	3	1	1													
33	5	3	11	4	3	3	2	2	1	1							
34	11	4	6	3	5	3	1	1									
35	11	4	6	3	5	3	3	3	1	1							
36	11	4	5	3	3	3	2	2	6	3	9	3	1	1			
37	10	4	1	1	4	4	2	2	7	3	3	3	5	3			

Appendix 4

Sample Output

```

network structure
weights represent arterial route travel time between node centers
node 1 (SPF) has outdegree 7
  arc to 2 (SBK) of weight 10.00
  arc to 3 (WJX) of weight 5.00
  arc to 4 (EJX) of weight 5.00
  arc to 5 (AVD) of weight 10.00
  arc to 6 (RVS) of weight 10.00
  arc to 8 (LMT) of weight 10.00
  arc to 9 (OWY) of weight 10.00
node 2 (SBK) has outdegree 7
  arc to 1 (SPF) of weight 10.00
  arc to 4 (EJX) of weight 10.00
  arc to 10 (ARL) of weight 10.00
  arc to 14 (SSD) of weight 15.00
  arc to 15 (JTB) of weight 10.00
  arc to 17 (MND) of weight 20.00
  arc to 5 (AVD) of weight 10.00
node 3 (WJX) has outdegree 6
  arc to 1 (SPF) of weight 5.00
  arc to 19 (WSD) of weight 20.00
  arc to 20 (MWH) of weight 20.00
  arc to 7 (PKV) of weight 5.00
  arc to 23 (CAL) of weight 20.00
  arc to 8 (LMT) of weight 10.00
node 4 (EJX) has outdegree 3
  arc to 1 (SPF) of weight 5.00
  arc to 10 (ARL) of weight 5.00
  arc to 2 (SBK) of weight 10.00
node 5 (AVD) has outdegree 6
  arc to 1 (SPF) of weight 10.00
  arc to 2 (SBK) of weight 10.00
  arc to 6 (RVS) of weight 5.00
  arc to 18 (TMQ) of weight 10.00
  arc to 19 (WSD) of weight 10.00
  arc to 20 (MWH) of weight 10.00
node 6 (RVS) has outdegree 2
  arc to 1 (SPF) of weight 10.00
  arc to 5 (AVD) of weight 5.00
node 7 (PKV) has outdegree 4
  arc to 3 (WJX) of weight 5.00
  arc to 19 (WSD) of weight 15.00
  arc to 20 (MWH) of weight 15.00
  arc to 8 (LMT) of weight 10.00
node 8 (LMT) has outdegree 5
  arc to 1 (SPF) of weight 10.00
  arc to 3 (WJX) of weight 10.00
  arc to 7 (PKV) of weight 10.00
  arc to 23 (CAL) of weight 20.00
  arc to 9 (OWY) of weight 10.00
node 9 (OWY) has outdegree 4
  arc to 1 (SPF) of weight 10.00

```

```

    arc to 8 (LMT) of weight 10.00
    arc to 25 (YUL) of weight 20.00
    arc to 11 (RGY) of weight 15.00
node 10 (ARL) has outdegree 4
    arc to 4 (EJX) of weight 5.00
    arc to 11 (RGY) of weight 5.00
    arc to 14 (SSD) of weight 10.00
    arc to 2 (SBK) of weight 10.00
node 11 (RGY) has outdegree 5
    arc to 10 (ARL) of weight 5.00
    arc to 9 (OWY) of weight 15.00
    arc to 12 (ATB) of weight 20.00
    arc to 15 (JTB) of weight 15.00
    arc to 14 (SSD) of weight 10.00
node 12 (ATB) has outdegree 3
    arc to 11 (RGY) of weight 20.00
    arc to 16 (SPL) of weight 10.00
    arc to 13 (JXB) of weight 10.00
node 13 (JXB) has outdegree 4
    arc to 12 (ATB) of weight 10.00
    arc to 16 (SPL) of weight 5.00
    arc to 32 (PVB) of weight 10.00
    arc to 15 (JTB) of weight 15.00
node 14 (SSD) has outdegree 5
    arc to 2 (SBK) of weight 15.00
    arc to 10 (ARL) of weight 10.00
    arc to 11 (RGY) of weight 10.00
    arc to 16 (SPL) of weight 15.00
    arc to 15 (JTB) of weight 7.50
node 15 (JTB) has outdegree 5
    arc to 2 (SBK) of weight 10.00
    arc to 14 (SSD) of weight 7.50
    arc to 11 (RGY) of weight 15.00
    arc to 17 (MND) of weight 10.00
    arc to 13 (JXB) of weight 15.00
node 16 (SPL) has outdegree 3
    arc to 14 (SSD) of weight 15.00
    arc to 12 (ATB) of weight 10.00
    arc to 13 (JXB) of weight 5.00
node 17 (MND) has outdegree 4
    arc to 2 (SBK) of weight 20.00
    arc to 15 (JTB) of weight 10.00
    arc to 33 (SWZ) of weight 20.00
    arc to 27 (OPE) of weight 20.00
node 18 (TMQ) has outdegree 3
    arc to 19 (WSD) of weight 10.00
    arc to 5 (AVD) of weight 10.00
    arc to 27 (OPE) of weight 15.00
node 19 (WSD) has outdegree 6
    arc to 28 (OPW) of weight 15.00
    arc to 20 (MWH) of weight 15.00
    arc to 7 (PKV) of weight 15.00
    arc to 3 (WJX) of weight 20.00
    arc to 5 (AVD) of weight 10.00
    arc to 18 (TMQ) of weight 10.00
node 20 (MWH) has outdegree 6
    arc to 19 (WSD) of weight 15.00
    arc to 5 (AVD) of weight 10.00
    arc to 3 (WJX) of weight 20.00
    arc to 7 (PKV) of weight 15.00
    arc to 22 (BRY) of weight 15.00
    arc to 21 (MCL) of weight 20.00
node 21 (MCL) has outdegree 1

```

```

    arc to 20 (MWH) of weight 20.00
node 22 (BRY) has outdegree 2
    arc to 20 (MWH) of weight 15.00
    arc to 23 (CAL) of weight 20.00
node 23 (CAL) has outdegree 5
    arc to 22 (BRY) of weight 20.00
    arc to 24 (HIL) of weight 15.00
    arc to 25 (YUL) of weight 20.00
    arc to 8 (LMT) of weight 20.00
    arc to 3 (WJX) of weight 20.00
node 24 (HIL) has outdegree 1
    arc to 23 (CAL) of weight 15.00
node 25 (YUL) has outdegree 4
    arc to 23 (CAL) of weight 20.00
    arc to 9 (OWY) of weight 20.00
    arc to 26 (FDB) of weight 20.00
    arc to 37 (KGB) of weight 25.00
node 26 (FDB) has outdegree 1
    arc to 25 (YUL) of weight 20.00
node 27 (OPE) has outdegree 4
    arc to 17 (MND) of weight 20.00
    arc to 18 (TMQ) of weight 15.00
    arc to 29 (GCS) of weight 15.00
    arc to 28 (OPW) of weight 10.00
node 28 (OPW) has outdegree 3
    arc to 19 (WSD) of weight 15.00
    arc to 27 (OPE) of weight 10.00
    arc to 30 (MBG) of weight 20.00
node 29 (GCS) has outdegree 4
    arc to 27 (OPE) of weight 15.00
    arc to 30 (MBG) of weight 20.00
    arc to 31 (PLK) of weight 30.00
    arc to 33 (SWZ) of weight 20.00
node 30 (MBG) has outdegree 2
    arc to 28 (OPW) of weight 20.00
    arc to 29 (GCS) of weight 20.00
node 31 (PLK) has outdegree 2
    arc to 29 (GCS) of weight 30.00
    arc to 36 (HST) of weight 20.00
node 32 (PVB) has outdegree 2
    arc to 13 (JXB) of weight 10.00
    arc to 34 (STA) of weight 35.00
node 33 (SWZ) has outdegree 4
    arc to 17 (MND) of weight 20.00
    arc to 29 (GCS) of weight 20.00
    arc to 34 (STA) of weight 35.00
    arc to 36 (HST) of weight 30.00
node 34 (STA) has outdegree 4
    arc to 33 (SWZ) of weight 35.00
    arc to 32 (PVB) of weight 35.00
    arc to 35 (CRB) of weight 20.00
    arc to 36 (HST) of weight 25.00
node 35 (CRB) has outdegree 2
    arc to 34 (STA) of weight 20.00
    arc to 36 (HST) of weight 20.00
node 36 (HST) has outdegree 2
    arc to 35 (CRB) of weight 20.00
    arc to 33 (SWZ) of weight 30.00
node 37 (KGB) has outdegree 1
    arc to 25 (YUL) of weight 25.00

```

hospital locations
 UMC, level 1 , 5 beds, in SPF

BAP, level 2 , 3 beds, in SBK
 MMC, level 2a, 2 beds, in SSD
 MTH, level 3 , 1 beds, in SPF
 STL, level 2a, 1 beds, in JTB
 BCH, level 2a, 1 beds, in JXB
 STV, level 2a, 2 beds, in RVS
 RVS, level 3 , 1 beds, in RVS
 HUM, level 2a, 1 beds, in OPW
 NGH, level 3 , 1 beds, in FDB
 FLG, level 3 , 1 beds, in STA
 PCH, level 3 , 1 beds, in PLK

ambulance call list

SPF will request these ambulances
 D9 from SPF with mean travel time 9.26 min
 D4 from SPF with mean travel time 9.26 min
 D7 from WJX with mean travel time 10.00 min
 D1 from EJX with mean travel time 10.00 min
 H1 from SPF with mean travel time 5.71 min
 H2 from SBK with mean travel time 4.74 min
 SBK will request these ambulances
 D13 from SBK with mean travel time 11.02 min
 D4 from SPF with mean travel time 14.14 min
 D9 from SPF with mean travel time 14.14 min
 D1 from EJX with mean travel time 14.14 min
 D23 from AVD with mean travel time 14.14 min
 D19 from ARL with mean travel time 14.14 min
 D28 from JTB with mean travel time 14.14 min
 H1 from SPF with mean travel time 4.74 min
 H2 from SBK with mean travel time 6.43 min
 WJX will request these ambulances
 D7 from WJX with mean travel time 11.55 min
 D4 from SPF with mean travel time 10.00 min
 D9 from SPF with mean travel time 10.00 min
 D26 from PKV with mean travel time 10.00 min
 H1 from SPF with mean travel time 4.30 min
 H2 from SBK with mean travel time 4.85 min
 EJX will request these ambulances
 D1 from EJX with mean travel time 8.16 min
 D4 from SPF with mean travel time 10.00 min
 D9 from SPF with mean travel time 10.00 min
 D19 from ARL with mean travel time 10.00 min
 H1 from SPF with mean travel time 4.18 min
 H2 from SBK with mean travel time 4.72 min
 AVD will request these ambulances
 D23 from AVD with mean travel time 9.57 min
 D10 from RVS with mean travel time 10.00 min
 D4 from SPF with mean travel time 14.14 min
 D9 from SPF with mean travel time 14.14 min
 D13 from SBK with mean travel time 14.14 min
 D25 from TMQ with mean travel time 14.14 min
 D22 from WSD with mean travel time 14.14 min
 D32 from MWH with mean travel time 14.14 min
 H1 from SPF with mean travel time 4.77 min
 H2 from SBK with mean travel time 4.54 min
 RVS will request these ambulances
 D10 from RVS with mean travel time 8.66 min
 D23 from AVD with mean travel time 10.00 min
 D4 from SPF with mean travel time 14.14 min
 D9 from SPF with mean travel time 14.14 min
 H1 from SPF with mean travel time 4.98 min
 H2 from SBK with mean travel time 4.49 min
 PKV will request these ambulances

D26	from PKV with mean travel time	10.61 min
D7	from WJX with mean travel time	10.00 min
D4	from SPF with mean travel time	14.14 min
D9	from SPF with mean travel time	14.14 min
D24	from LMT with mean travel time	14.14 min
H1	from SPF with mean travel time	5.00 min
H2	from SBK with mean travel time	5.53 min
LMT	will request these ambulances	
D24	from LMT with mean travel time	10.95 min
D4	from SPF with mean travel time	14.14 min
D9	from SPF with mean travel time	14.14 min
D7	from WJX with mean travel time	14.14 min
D26	from PKV with mean travel time	14.14 min
D35	from OWY with mean travel time	14.14 min
H1	from SPF with mean travel time	5.67 min
H2	from SBK with mean travel time	6.41 min
OWY	will request these ambulances	
D35	from OWY with mean travel time	11.73 min
D4	from SPF with mean travel time	14.14 min
D9	from SPF with mean travel time	14.14 min
D24	from LMT with mean travel time	14.14 min
H1	from SPF with mean travel time	6.87 min
H2	from SBK with mean travel time	7.58 min
ARL	will request these ambulances	
D19	from ARL with mean travel time	8.66 min
D1	from EJX with mean travel time	10.00 min
D30	from RGY with mean travel time	10.00 min
H1	from SPF with mean travel time	4.66 min
H2	from SBK with mean travel time	4.65 min
RGY	will request these ambulances	
D30	from RGY with mean travel time	11.40 min
D19	from ARL with mean travel time	10.00 min
D1	from EJX with mean travel time	14.14 min
D20	from SSD with mean travel time	14.14 min
H1	from SPF with mean travel time	5.17 min
H2	from SBK with mean travel time	4.80 min
ATB	will request these ambulances	
D41	from ATB with mean travel time	11.55 min
D71	from JXB with mean travel time	14.14 min
D50	from SPL with mean travel time	14.14 min
H1	from SPF with mean travel time	6.70 min
H2	from SBK with mean travel time	6.37 min
JXB	will request these ambulances	
D71	from JXB with mean travel time	10.00 min
D50	from SPL with mean travel time	10.00 min
D41	from ATB with mean travel time	14.14 min
S81	from PVB with mean travel time	14.14 min
H1	from SPF with mean travel time	7.04 min
H2	from SBK with mean travel time	6.47 min
SSD	will request these ambulances	
D20	from SSD with mean travel time	10.72 min
D28	from JTB with mean travel time	12.25 min
D19	from ARL with mean travel time	14.14 min
D30	from RGY with mean travel time	14.14 min
H1	from SPF with mean travel time	5.53 min
H2	from SBK with mean travel time	4.81 min
JTB	will request these ambulances	
D28	from JTB with mean travel time	10.72 min
D20	from SSD with mean travel time	12.25 min
D13	from SBK with mean travel time	14.14 min
D42	from MND with mean travel time	14.14 min
H1	from SPF with mean travel time	6.44 min
H2	from SBK with mean travel time	5.69 min

SPL	will	request these ambulances	
D50	from SPL	with mean travel time	10.00 min
D71	from JXB	with mean travel time	10.00 min
D41	from ATB	with mean travel time	14.14 min
H1	from SPF	with mean travel time	6.60 min
H2	from SBK	with mean travel time	6.01 min
MND	will	request these ambulances	
D42	from MND	with mean travel time	13.23 min
D28	from JTB	with mean travel time	14.14 min
D20	from SSD	with mean travel time	22.50 min
H1	from SPF	with mean travel time	8.26 min
H2	from SBK	with mean travel time	7.53 min
TMQ	will	request these ambulances	
D25	from TMQ	with mean travel time	10.80 min
D23	from AVD	with mean travel time	14.14 min
D22	from WSD	with mean travel time	14.14 min
H1	from SPF	with mean travel time	6.45 min
H2	from SBK	with mean travel time	5.83 min
WSD	will	request these ambulances	
D22	from WSD	with mean travel time	11.90 min
D23	from AVD	with mean travel time	14.14 min
D25	from TMQ	with mean travel time	14.14 min
H1	from SPF	with mean travel time	6.57 min
H2	from SBK	with mean travel time	6.01 min
MWH	will	request these ambulances	
D32	from MWH	with mean travel time	12.58 min
D23	from AVD	with mean travel time	14.14 min
D10	from RVS	with mean travel time	20.00 min
D26	from PKV	with mean travel time	20.00 min
D22	from WSD	with mean travel time	20.00 min
N6	from BRY	with mean travel time	20.00 min
H1	from SPF	with mean travel time	5.67 min
H2	from SBK	with mean travel time	5.64 min
MCL	will	request these ambulances	
B32	from MCL	with mean travel time	14.14 min
D32	from MWH	with mean travel time	25.00 min
D23	from AVD	with mean travel time	35.00 min
H1	from SPF	with mean travel time	10.04 min
H2	from SBK	with mean travel time	9.77 min
BRY	will	request these ambulances	
N6	from BRY	with mean travel time	13.23 min
D32	from MWH	with mean travel time	20.00 min
N5	from CAL	with mean travel time	25.00 min
H1	from SPF	with mean travel time	8.18 min
H2	from SBK	with mean travel time	8.40 min
CAL	will	request these ambulances	
N5	from CAL	with mean travel time	13.78 min
N4	from HIL	with mean travel time	20.00 min
D7	from WJX	with mean travel time	25.00 min
D24	from LMT	with mean travel time	25.00 min
N6	from BRY	with mean travel time	25.00 min
N3	from YUL	with mean travel time	25.00 min
H1	from SPF	with mean travel time	9.87 min
H2	from SBK	with mean travel time	10.51 min
HIL	will	request these ambulances	
N4	from HIL	with mean travel time	12.25 min
N5	from CAL	with mean travel time	20.00 min
D7	from WJX	with mean travel time	40.00 min
D24	from LMT	with mean travel time	40.00 min
N6	from BRY	with mean travel time	40.00 min
N3	from YUL	with mean travel time	40.00 min
H1	from SPF	with mean travel time	12.99 min
H2	from SBK	with mean travel time	13.64 min

YUL	will request these ambulances	
N3	from YUL with mean travel time	15.62 min
D35	from OWY with mean travel time	25.00 min
N5	from CAL with mean travel time	25.00 min
N1	from FDB with mean travel time	25.00 min
N2	from FDB with mean travel time	25.00 min
H1	from SPF with mean travel time	10.79 min
H2	from SBK with mean travel time	11.49 min
FDB	will request these ambulances	
N2	from FDB with mean travel time	14.14 min
N1	from FDB with mean travel time	14.14 min
N3	from YUL with mean travel time	25.00 min
H1	from SPF with mean travel time	12.08 min
H2	from SBK with mean travel time	12.70 min
OPE	will request these ambulances	
C72	from OPE with mean travel time	12.25 min
C76	from OPW with mean travel time	14.14 min
D25	from TMQ with mean travel time	20.00 min
C70	from GCS with mean travel time	20.00 min
H1	from SPF with mean travel time	8.45 min
H2	from SBK with mean travel time	7.76 min
OPW	will request these ambulances	
C76	from OPW with mean travel time	12.25 min
C72	from OPE with mean travel time	14.14 min
D22	from WSD with mean travel time	20.00 min
H1	from SPF with mean travel time	8.12 min
H2	from SBK with mean travel time	7.44 min
GCS	will request these ambulances	
C70	from GCS with mean travel time	15.62 min
C72	from OPE with mean travel time	20.00 min
C74	from MBG with mean travel time	25.00 min
S82	from SWZ with mean travel time	25.00 min
H1	from SPF with mean travel time	12.35 min
H2	from SBK with mean travel time	11.62 min
MBG	will request these ambulances	
C74	from MBG with mean travel time	14.14 min
C76	from OPW with mean travel time	25.00 min
C70	from GCS with mean travel time	25.00 min
H1	from SPF with mean travel time	11.32 min
H2	from SBK with mean travel time	10.67 min
PLK	will request these ambulances	
P1	from PLK with mean travel time	17.50 min
S88	from HST with mean travel time	85.00 min
C70	from GCS with mean travel time	35.00 min
H1	from SPF with mean travel time	20.78 min
H2	from SBK with mean travel time	20.06 min
PVB	will request these ambulances	
S81	from PVB with mean travel time	16.25 min
D71	from JXB with mean travel time	14.14 min
D50	from SPL with mean travel time	20.00 min
H1	from SPF with mean travel time	7.82 min
H2	from SBK with mean travel time	7.10 min
SWZ	will request these ambulances	
S82	from SWZ with mean travel time	18.12 min
D42	from MND with mean travel time	25.00 min
C70	from GCS with mean travel time	25.00 min
H1	from SPF with mean travel time	10.10 min
H2	from SBK with mean travel time	9.33 min
STA	will request these ambulances	
S83	from STA with mean travel time	19.37 min
S80	from STA with mean travel time	19.37 min
S88	from HST with mean travel time	45.00 min
H1	from SPF with mean travel time	16.00 min

	H2	from SBK with mean travel time	15.21 min
CRB	will	request these ambulances	
	S80	from STA with mean travel time	25.00 min
	S83	from STA with mean travel time	25.00 min
	S88	from HST with mean travel time	25.00 min
	H1	from SPF with mean travel time	17.32 min
	H2	from SBK with mean travel time	16.49 min
HST	will	request these ambulances	
	S88	from HST with mean travel time	17.50 min
	S82	from SWZ with mean travel time	35.00 min
	S80	from STA with mean travel time	30.00 min
	S83	from STA with mean travel time	30.00 min
	H1	from SPF with mean travel time	17.46 min
	H2	from SBK with mean travel time	16.72 min
KGB	will	request these ambulances	
	G1	from KGB with mean travel time	17.50 min
	N3	from YUL with mean travel time	30.00 min
	D35	from OWY with mean travel time	50.00 min
	N5	from CAL with mean travel time	50.00 min
	N1	from FDB with mean travel time	50.00 min
	N2	from FDB with mean travel time	50.00 min
	H1	from SPF with mean travel time	14.69 min
	H2	from SBK with mean travel time	15.38 min

hospital dispatch list

SPF	victims will go to these hospitals	
	MTH, level 3 , in SPF, with mean travel time	9.26 min
	UMC, level 1 , in SPF, with mean travel time	9.26 min
	BAP, level 2 , in SBK, with mean travel time	14.14 min
	STV, level 2a, in RVS, with mean travel time	14.14 min
SBK	victims will go to these hospitals	
	BAP, level 2 , in SBK, with mean travel time	11.02 min
	UMC, level 1 , in SPF, with mean travel time	14.14 min
	MTH, level 3 , in SPF, with mean travel time	14.14 min
	STL, level 2a, in JTB, with mean travel time	14.14 min
WJX	victims will go to these hospitals	
	UMC, level 1 , in SPF, with mean travel time	10.00 min
	MTH, level 3 , in SPF, with mean travel time	10.00 min
	BAP, level 2 , in SBK, with mean travel time	20.00 min
	STV, level 2a, in RVS, with mean travel time	20.00 min
EJX	victims will go to these hospitals	
	UMC, level 1 , in SPF, with mean travel time	10.00 min
	MTH, level 3 , in SPF, with mean travel time	10.00 min
	BAP, level 2 , in SBK, with mean travel time	14.14 min
AVD	victims will go to these hospitals	
	STV, level 2a, in RVS, with mean travel time	10.00 min
	RVS, level 3 , in RVS, with mean travel time	10.00 min
	UMC, level 1 , in SPF, with mean travel time	14.14 min
RVS	victims will go to these hospitals	
	RVS, level 3 , in RVS, with mean travel time	8.66 min
	STV, level 2a, in RVS, with mean travel time	8.66 min
	UMC, level 1 , in SPF, with mean travel time	14.14 min
PKV	victims will go to these hospitals	
	UMC, level 1 , in SPF, with mean travel time	14.14 min
	MTH, level 3 , in SPF, with mean travel time	14.14 min
	BAP, level 2 , in SBK, with mean travel time	25.00 min
	STV, level 2a, in RVS, with mean travel time	25.00 min
LMT	victims will go to these hospitals	
	UMC, level 1 , in SPF, with mean travel time	14.14 min
	MTH, level 3 , in SPF, with mean travel time	14.14 min
	BAP, level 2 , in SBK, with mean travel time	25.00 min
	STV, level 2a, in RVS, with mean travel time	25.00 min
OWY	victims will go to these hospitals	

	UMC, level 1 , in SPF, with mean travel time	14.14 min
	MTH, level 3 , in SPF, with mean travel time	14.14 min
	BAP, level 2 , in SBK, with mean travel time	25.00 min
	STV, level 2a, in RVS, with mean travel time	25.00 min
	MMC, level 2a, in SSD, with mean travel time	30.00 min
ARL	victims will go to these hospitals	
	UMC, level 1 , in SPF, with mean travel time	14.14 min
	MTH, level 3 , in SPF, with mean travel time	14.14 min
	BAP, level 2 , in SBK, with mean travel time	14.14 min
	MMC, level 2a, in SSD, with mean travel time	14.14 min
RGY	victims will go to these hospitals	
	MMC, level 2a, in SSD, with mean travel time	14.14 min
	UMC, level 1 , in SPF, with mean travel time	20.00 min
ATB	victims will go to these hospitals	
	BCH, level 2a, in JXB, with mean travel time	14.14 min
	UMC, level 1 , in SPF, with mean travel time	40.00 min
JXB	victims will go to these hospitals	
	BCH, level 2a, in JXB, with mean travel time	10.00 min
	UMC, level 1 , in SPF, with mean travel time	40.00 min
SSD	victims will go to these hospitals	
	MMC, level 2a, in SSD, with mean travel time	10.72 min
	STL, level 2a, in JTB, with mean travel time	12.25 min
	UMC, level 1 , in SPF, with mean travel time	25.00 min
JTB	victims will go to these hospitals	
	STL, level 2a, in JTB, with mean travel time	10.72 min
	MMC, level 2a, in SSD, with mean travel time	12.25 min
	UMC, level 1 , in SPF, with mean travel time	25.00 min
SPL	victims will go to these hospitals	
	BCH, level 2a, in JXB, with mean travel time	10.00 min
	UMC, level 1 , in SPF, with mean travel time	40.00 min
MND	victims will go to these hospitals	
	STL, level 2a, in JTB, with mean travel time	14.14 min
	UMC, level 1 , in SPF, with mean travel time	35.00 min
TMQ	victims will go to these hospitals	
	STV, level 2a, in RVS, with mean travel time	20.00 min
	RVS, level 3 , in RVS, with mean travel time	20.00 min
	UMC, level 1 , in SPF, with mean travel time	25.00 min
	MTH, level 3 , in SPF, with mean travel time	25.00 min
	BAP, level 2 , in SBK, with mean travel time	25.00 min
WSD	victims will go to these hospitals	
	STV, level 2a, in RVS, with mean travel time	20.00 min
	RVS, level 3 , in RVS, with mean travel time	20.00 min
	HUM, level 2a, in OPW, with mean travel time	20.00 min
	UMC, level 1 , in SPF, with mean travel time	25.00 min
	MTH, level 3 , in SPF, with mean travel time	25.00 min
	BAP, level 2 , in SBK, with mean travel time	25.00 min
MWH	victims will go to these hospitals	
	STV, level 2a, in RVS, with mean travel time	20.00 min
	RVS, level 3 , in RVS, with mean travel time	20.00 min
	UMC, level 1 , in SPF, with mean travel time	25.00 min
	MTH, level 3 , in SPF, with mean travel time	25.00 min
	BAP, level 2 , in SBK, with mean travel time	25.00 min
MCL	victims will go to these hospitals	
	STV, level 2a, in RVS, with mean travel time	40.00 min
	RVS, level 3 , in RVS, with mean travel time	40.00 min
	UMC, level 1 , in SPF, with mean travel time	45.00 min
	MTH, level 3 , in SPF, with mean travel time	45.00 min
	BAP, level 2 , in SBK, with mean travel time	45.00 min
BRY	victims will go to these hospitals	
	STV, level 2a, in RVS, with mean travel time	35.00 min
	RVS, level 3 , in RVS, with mean travel time	35.00 min
	UMC, level 1 , in SPF, with mean travel time	40.00 min
	MTH, level 3 , in SPF, with mean travel time	40.00 min

	BAP, level 2 , in SBK, with mean travel time	40.00 min
CAL	victims will go to these hospitals	
	UMC, level 1 , in SPF, with mean travel time	30.00 min
	MTH, level 3 , in SPF, with mean travel time	30.00 min
	BAP, level 2 , in SBK, with mean travel time	40.00 min
	STV, level 2a, in RVS, with mean travel time	40.00 min
	MMC, level 2a, in SSD, with mean travel time	50.00 min
	STL, level 2a, in JTB, with mean travel time	50.00 min
HIL	victims will go to these hospitals	
	UMC, level 1 , in SPF, with mean travel time	45.00 min
	MTH, level 3 , in SPF, with mean travel time	45.00 min
	BAP, level 2 , in SBK, with mean travel time	55.00 min
	STV, level 2a, in RVS, with mean travel time	55.00 min
	RVS, level 3 , in RVS, with mean travel time	55.00 min
	MMC, level 2a, in SSD, with mean travel time	65.00 min
	STL, level 2a, in JTB, with mean travel time	65.00 min
YUL	victims will go to these hospitals	
	NGH, level 3 , in FDB, with mean travel time	25.00 min
	UMC, level 1 , in SPF, with mean travel time	35.00 min
	BAP, level 2 , in SBK, with mean travel time	45.00 min
	STV, level 2a, in RVS, with mean travel time	45.00 min
	MMC, level 2a, in SSD, with mean travel time	50.00 min
	STL, level 2a, in JTB, with mean travel time	55.00 min
FDB	victims will go to these hospitals	
	NGH, level 3 , in FDB, with mean travel time	14.14 min
	UMC, level 1 , in SPF, with mean travel time	55.00 min
	BAP, level 2 , in SBK, with mean travel time	65.00 min
	STV, level 2a, in RVS, with mean travel time	65.00 min
	MMC, level 2a, in SSD, with mean travel time	70.00 min
	STL, level 2a, in JTB, with mean travel time	75.00 min
OPE	victims will go to these hospitals	
	HUM, level 2a, in OPW, with mean travel time	14.14 min
	UMC, level 1 , in SPF, with mean travel time	40.00 min
OPW	victims will go to these hospitals	
	HUM, level 2a, in OPW, with mean travel time	12.25 min
	UMC, level 1 , in SPF, with mean travel time	40.00 min
GCS	victims will go to these hospitals	
	HUM, level 2a, in OPW, with mean travel time	30.00 min
	PCH, level 3 , in PLK, with mean travel time	35.00 min
	UMC, level 1 , in SPF, with mean travel time	55.00 min
MBG	victims will go to these hospitals	
	HUM, level 2a, in OPW, with mean travel time	25.00 min
	UMC, level 1 , in SPF, with mean travel time	60.00 min
PLK	victims will go to these hospitals	
	PCH, level 3 , in PLK, with mean travel time	17.50 min
	HUM, level 2a, in OPW, with mean travel time	60.00 min
	UMC, level 1 , in SPF, with mean travel time	85.00 min
PVB	victims will go to these hospitals	
	BCH, level 2a, in JXB, with mean travel time	14.14 min
	UMC, level 1 , in SPF, with mean travel time	50.00 min
SWZ	victims will go to these hospitals	
	STL, level 2a, in JTB, with mean travel time	35.00 min
	FLG, level 3 , in STA, with mean travel time	40.00 min
	MMC, level 2a, in SSD, with mean travel time	42.50 min
	BAP, level 2 , in SBK, with mean travel time	45.00 min
	UMC, level 1 , in SPF, with mean travel time	55.00 min
STA	victims will go to these hospitals	
	FLG, level 3 , in STA, with mean travel time	19.37 min
	BCH, level 2a, in JXB, with mean travel time	50.00 min
	STL, level 2a, in JTB, with mean travel time	65.00 min
	UMC, level 1 , in SPF, with mean travel time	85.00 min
CRB	victims will go to these hospitals	
	FLG, level 3 , in STA, with mean travel time	25.00 min

BCH, level 2a, in JXB, with mean travel time	70.00 min
STL, level 2a, in JTB, with mean travel time	85.00 min
MMC, level 2a, in SSD, with mean travel time	90.00 min
UMC, level 1, in SPF, with mean travel time	105.00 min
HST victims will go to these hospitals	
FLG, level 3, in STA, with mean travel time	45.00 min
STL, level 2a, in JTB, with mean travel time	65.00 min
MMC, level 2a, in SSD, with mean travel time	72.50 min
BAP, level 2, in SBK, with mean travel time	75.00 min
BCH, level 2a, in JXB, with mean travel time	80.00 min
HUM, level 2a, in OPW, with mean travel time	80.00 min
UMC, level 1, in SPF, with mean travel time	85.00 min
KGB victims will go to these hospitals	
NGH, level 3, in FDB, with mean travel time	50.00 min
UMC, level 1, in SPF, with mean travel time	60.00 min
MTH, level 3, in SPF, with mean travel time	60.00 min
BAP, level 2, in SBK, with mean travel time	70.00 min
STV, level 2a, in RVS, with mean travel time	70.00 min
MMC, level 2a, in SSD, with mean travel time	75.00 min
STL, level 2a, in JTB, with mean travel time	80.00 min

Triage rule:

Champ TS <= 7.1 goes to level 1, > 7.4 may go to level 3

Travel times exceeding minimum time by less than tolerance included in
routine dispatch lists

hospital choice tolerance 30%

ambulance choice tolerance 35%

```
run 1
duration      12.20 hours    no.accs      9      no. tr. pts  12
% blunt      55.56
amb. utilization .041
helo utilization .045
diverts      0
overrides    0
deaths       2
est deaths   .59
queue for amb mean 0.      max 0
no.accs pended: 0
av pending time: 0.      minutes

hospital utilization
name      avg tr load    max tr load    reserve    % red time
UMC      .258          3          1.000      0.
BAP      0.          0          1.000      0.
MMC      0.          0          1.000      0.
MTH      .023          1          .977      0.
STL      0.          0          1.000      0.
BCH      0.          0          1.000      0.
STV      0.          0          1.000      0.
RVS      0.          0          1.000      0.
HUM      .025          1          .975      0.
NGH      0.          0          1.000      0.
FLG      0.          0          1.000      0.
PCH      0.          0          1.000      0.
```

```
run 2
duration      18.58 hours    no.accs     21      no. tr. pts  31
% blunt      52.38
amb. utilization .090
```

helo utilization .156
 diverts 0
 overrides 0
 deaths 6
 est deaths 2.15
 queue for amb mean 0. max 0
 no.accs pended: 0
 av pending time: 0. minutes

hospital utilization				
name	avg tr load	max tr load	reserve	% red time
UMC	.270	4	1.000	0.
BAP	.038	1	1.000	0.
MMC	0.	0	1.000	0.
MTH	0.	0	1.000	0.
STL	.031	1	.969	0.
BCH	.023	1	.977	0.
STV	.038	2	.989	0.
RVS	.025	1	.975	0.
HUM	0.	0	1.000	0.
NGH	0.	0	1.000	0.
FLG	.024	1	.976	0.
PCH	0.	0	1.000	0.

run 3
 duration 13.50 hours no.accs 12 no. tr. pts 20
 % blunt 83.33
 amb. utilization .067
 helo utilization .087
 diverts 0
 overrides 0
 deaths 4
 est deaths 1.20
 queue for amb mean .079 max 1
 no.accs pended: 1
 av pending time: 64.258 minutes

hospital utilization				
name	avg tr load	max tr load	reserve	% red time
UMC	.355	4	1.000	0.
BAP	0.	0	1.000	0.
MMC	0.	0	1.000	0.
MTH	.024	1	.976	0.
STL	.079	2	.952	7.4
BCH	0.	0	1.000	0.
STV	.086	1	1.000	0.
RVS	0.	0	1.000	0.
HUM	0.	0	1.000	0.
NGH	0.	0	1.000	0.
FLG	0.	0	1.000	0.
PCH	0.	0	1.000	0.

run 4
 duration 9.81 hours no.accs 4 no. tr. pts 3
 % blunt 50.00
 amb. utilization .018
 helo utilization 0.
 diverts 0
 overrides 0
 deaths 0
 est deaths .01

queue for amb mean 0. max 0
no.accs pended: 0
av pending time: 0. minutes

hospital utilization				
name	avg tr load	max tr load	reserve	% red time
UMC	.091	2	1.000	0.
BAP	0.	0	1.000	0.
MMC	0.	0	1.000	0.
MTH	0.	0	1.000	0.
STL	0.	0	1.000	0.
BCH	0.	0	1.000	0.
STV	0.	0	1.000	0.
RVS	0.	0	1.000	0.
HUM	0.	0	1.000	0.
NGH	0.	0	1.000	0.
FLG	0.	0	1.000	0.
PCH	0.	0	1.000	0.

Results after 4 runs of at least .25 days per run

	average	sd (of runwise means)	max
duration	.56	.154	.77
no. accidents	11.5	7.14	21
% blunt	60.9		
no. tr. patients	16.5	11.90	31
no. deaths	3.0		
no. est deaths	1.0		
amb. util	.10	.030	
amb. queue	.020	.0397	
		mean	.3
		glob	1

hospital utilization						
name	load		max		reserve	
	avg	sd	avg	global	avg	sd
UMC	.24	.110	3	4	1.000	0.
BAP	.01	.019	0	1	1.000	0.
MMC	0.	0.	0	0	1.000	0.
MTH	.01	.014	0	1	.988	.0135
STL	.03	.037	1	2	.980	.0241
BCH	.01	.011	0	1	.994	.0114
STV	.03	.041	1	2	.997	.0054
RVS	.01	.012	0	1	.994	.0123
HUM	.01	.013	0	1	.994	.0127
NGH	0.	0.	0	0	1.000	0.
FLG	.01	.012	0	1	.994	.0122
PCH	0.	0.	0	0	1.000	0.

Appendix 5

Fitting Input Distributions

Time to secure

20 of 215 needed securing:
model by beta(21, 196)

N OF CASE	20
MINIMUM	0.830
MAXIMUM	32.770
RANGE	31.940
MEAN	6.150
VARIANCE	46.505
STANDARD DEV	6.819
STD. ERROR	1.525
SKEWNESS(G1)	3.141
KURTOSIS(G2)	9.933
SUM	123.000
C.V.	1.109
MEDIAN	4.230

Figure 8. Distribution given need to secure.

Figure 9. Probability plot: exponential.

Figure 10. Quantile plot: exponential.

Figure 11. Probability plot: Weibull distribution.

Figure 12. Quantile plot: Weibull(6.686, 1.412).

Figure 13. Probability plot: lognormal.

Figure 14. Quantile plot: lognormal.

Time to patient found

N	215
MINIMUM	0.050
MAXIMUM	12.920
RANGE	12.870
MEAN	2.220
VARIANCE	5.818
STANDARD DEV	2.412
STD. ERROR	0.165
SKEWNESS (G1)	1.696
KURTOSIS (G2)	2.697
SUM	477.380
C.V.	1.086
MEDIAN	1.120

Figure 15. Distribution of time to patient.

Figure 16. Probability plot: exponential.

Figure 17. Quantile plot: exponential.

Exponential fits very well.

Time on scene

Total scene time for those needing extrication and those not were not significantly different (Mann-Whitney $P = .15$, Komolgorov-Smirnov $P = .11$).

N	185
MINIMUM	1.000
MAXIMUM	30.830
RANGE	29.830
MEAN	10.404
VARIANCE	34.368
STANDARD DEV	5.862
STD. ERROR	0.431
SKEWNESS (G1)	1.149
KURTOSIS (G2)	1.394
SUM	1924.780
C.V.	0.563
MEDIAN	9.070

Figure 18. Distribution of scene treatment time (includes extrication if needed).

Figure 19. Probability plot: Weibull.

Figure 20. Quantile plot: Weibull(11.023, 1.892).

Figure 21. Probability plot: gamma(3).

Figure 22. Quantile plot: gamma(3).

Figure 23. Weibull (dashed line) and gamma pdf's.

Time to hospital

N	184
MINIMUM	1.370
MAXIMUM	37.230
RANGE	35.860
MEAN	10.295
VARIANCE	29.925
STANDARD DEV	5.470
STD. ERROR	0.403
SKEWNESS (G1)	1.072
KURTOSIS (G2)	2.268
SUM	1894.250
C.V.	0.531
MEDIAN	9.775

Figure 24. Distribution of time to hospital.

Figure 25. Probability plot: Weibull.

Figure 26. Quantile plot: Weibull(9.974, 2.242).

Figure 27. Probability plot: lognormal.

Figure 28. Quantile plot: lognormal.

Figure 29. Probability plot: gamma(3).

Figure 30. Quantile plot: gamma(3).

Time to release of pt

N	177
MINIMUM	0.270
MAXIMUM	12.000
RANGE	11.730
MEAN	2.598
VARIANCE	3.788
STANDARD DEV	1.946
STD. ERROR	0.146
SKEWNESS (G1)	2.366
KURTOSIS (G2)	7.257
SUM	459.880
C.V.	0.749
MEDIAN	2.070

Figure 31. Distribution of time to release of patient.

Figure 32. Probability plot: Weibull.

Figure 33. Quantile plot: Weibull(2.718, 1.810).

Figure 34. Probability plot: lognormal.

Figure 35. Quantile plot: lognormal.

Figure 36. Probability plot: gamma.

Figure 37. Quantile plot: gamma(1.5).

Figure 38. Probability plot: exponential.

Figure 39. Quantile plot: exponential.

Appendix 6

Log of Model Assumptions

6.1 Distribution of Accidents

It is assumed that the average number of accidents per unit time is unchanging; i.e., there is no long term growth or decline in the level of demand. Only the number of injuring incidents is assumed to vary with time and location; severity of injury and type of injury are assumed to be independent. Similarly, it is assumed that there is no interaction or synergism between time and location.

6.2 Definitive Care Survival

There is no assumption that patients receiving definitive care in a Level 1 center have increased survival compared to those in other centers (after adjusting for severity of injury). There are published reports that this may not be true (that Level 1 patients do better), so this assumption may need to be examined critically.

6.3 Private Travel

The function expressing the probability of private travel as inversely proportional to severity and between .30 and .01 is entirely empiric, based on experience, plausibility, and scant data suggesting that overall about one-fifth of non-

trivial trauma patients find their own way to medical care. In addition, the model currently assumes that all patients involved in an accident make the same decision. This clearly plays a role in the real world model but is not completely valid. Other considerations, including proximity to a hospital and the actual activity producing the injury are also involved in as yet unspecified ways.

Appendix 7

Log of Improvements and Enhancements

7.1 Improvements

The following items (in no particular order) will enable the existing model to run more efficiently, more realistically, or be more easily maintained, but do not add additional function.

7.1.1 Transit time. The permanent entity ambulance currently maintains the attribute of transit time. This is more realistically an attribute of an ambulance run process than it is of an ambulance, and in keeping with the second principle of Section 4.2 should be moved to the process.

7.1.2 Choke points. Routines for building call and dispatch lists should take better account of the choke point variable than they do presently.

7.1.3 Events. Coding the respiratory support therapeutic interventions as events, rather than inserting them directly into either (or both) of the patient or ambulance run process routines allowed produced much cleaner, more easily maintained code. Other critical therapeutic maneuvers such as IV starting or blood transfusion should be recoded into this form.

7.1.4 End-of-run. The process of cancelling and then rescheduling the clear.reds event if another run follows the current should be changed to leave the event scheduled unless no runs are to follow.

7.1.5 Memory management. The duplicate representation of arcs in the current implementation requires large amounts of memory, leading to disk swapping and poor performance in large models. Since much of this information is redundant, and since it is infrequently referenced once the actual simulation begins, a more efficient representation should produce disproportionate benefits in run times.

7.2 Enhancements

The following items are additional (new) capabilities that would increase the utility of the model, but will be deferred at this time as they are not critical to the proof of concept.

7.2.1 Non-trauma patients. Information on medical patients handled by the system should be added to the model. In particular, some method of representing the "walk-in" medical load on emergency departments should be added as this is a major reason for a hospital's going on divert. For example, this could be done quite simply as a random external event, without having to explicitly model large numbers of "walk-in" patients.

7.2.2 Injury model. Better and more detailed models of injury such as ASCOT [Champion90] are now available.

Modification of the model to use ASCOT or a similar measure of injury severity might allow better prediction of outcome and identification of subsets of patients for whom special policies may be beneficial. The use of ASCOT, however, would require far more detailed epidemiologic information about injury patterns than is currently available, although it is being collected in the national Trauma Registry program of the American College of Surgeons (TRACS).

7.2.3 Transfers. While they are only a small fraction of the total volume of trauma patients, inter-hospital transfers are frequently a considerable source of contention. It would be desirable to model the transfer of patients among hospitals as representing an important aspect of the system, but it has proven extremely difficult to obtain reliable information, and what data is available is highly suspect as misleading at best.

7.2.4 Data editor. There is currently no support for creating the data files used to drive the simulation, nor is there any error checking for illogical or impossible conditions, e.g., a node with efferent but no afferent arcs. For complex models, creating the data files with a text can be tedious and prone to error; modification may be even more difficult. A data editor, particularly if it were able to graphically represent the transportation network, would make

the model much easier to use. Error checking could be provided along with the editing function to ensure that the files finally submitted to the model did not contain logical errors.

7.2.5 Graphical output. Although it may restrict portability somewhat, a graphical display of system activity may be useful in establishing face validity of the model.

7.2.6 Trace control. Currently, the trace output is "all or nothing," making it unwieldy when the region of interest lies deep in a long run or series of runs. The ability to turn the trace on or off from within the program would enhance the usefulness of the trace.

7.2.7 Interruption. Direct (paired) comparison of regenerative method simulation data must be done at comparable points in each experimental arm. The number and location of such points is unpredictable, and they become fewer and are spread farther apart as the number of arms in an experiment increases. It would therefore be advantageous to make model runs interruptible, so that if there were not sufficient convergence points for analysis after some period of simulated time, the model could be restarted at that point and run further forward. Since the simulation mechanism is regenerative, this can in principle be done manually simply by preserving the random number seeds (or choosing different random number generators altogether) and

starting another series of runs with the RNG's reset and time.v set to its value at the end of the previous corresponding run.

7.2.8 Non-regenerative Simulation. The difficulties of statistical analysis of the regenerative method data suggest that it might be advantageous to abandon this method in favor of traditional non-terminating simulation analysis of steady-state cycle parameters. This would require analysis of the startup transients which has not been performed for this model. It is not clear whether this would increase or decrease the required length of the simulation.

Appendix 8

Log of Program Bugs

8.1 Discrete-continuous Interaction

A variety of hard-to-diagnose problems sometimes appear when the discrete simulation portions of the program interact with the continuous simulation portions. For example, if the minimum step size for the integrator routine in the continuous simulation modules is too large, it is at least theoretically possible for the two sections to become unsynchronized. While it is felt that most of the areas at risk have been protected by a combination of local code and by a policy of keeping the maximum step size small, confidence in the model's reliability would be significantly enhanced by systematically eliminating potential areas of interaction.

8.2 Pended Accidents

The dispatcher currently only checks the first accident in the pending set when a new ambulance becomes available; the entire set should be checked. Since, under the conditions modeled for northeast Florida, the pending set virtually never contains more than one accident, this error was not initially apparent.

Vita

Robert L. Wears has BA and MD degrees from the Johns Hopkins University (1969 and 1973, respectively). He is currently Associate Professor in the Department of Surgery, Division of Emergency Medicine at the University of Florida Health Science Center in Jacksonville, and expects to receive an MS in Computer and Information Sciences from the University of North Florida on April 30, 1993.

Robert has extensive interests in the application of computer-based modeling and analysis techniques to clinical and health policy problems, and has published several examples of their use in the medical literature with Dr. Charles N. Winton, his thesis advisor. His work includes statistical and connectionist approaches to computer modeling, with extensive experience in C, SYSTAT, and SIMSCRIPT. He has developed a fellowship in medical informatics for graduate physicians, installed the first LAN at UFHSC-Jax, and is active in promoting the application of modern information methods in the clinical setting.

Dr. Wears has been married to his first wife for 21 years, and has two teenage children, a dog, a fish, and is a raving windsurfer.