Mars Analyst's Guide

Mars Simulation Infrastructure Library, V1.33

September 2010

William H. Duquette

William.H.Duquette@jpl.nasa.gov

Jet Propulsion Laboratory

Table of Contents

1. INTRODUCTION

This document presents the models and related constructs implemented by version 1 of the Mars Simulation Infrastructure Library (Mars). The models are described in sufficient detail to allow implementation; the implementation itself is not in the scope of this document.

1.1 Other Mars Documents

The MARS documentation set may be found in the “mars/docs” directory of the Mars build tree; open “mars/docs/index.html” in a web browser, and follow the links. The documentation is usually included in the documentation set for client simulations. Otherwise, documents can be obtained directly from the JNEM or Athena projects; contact David.R.Hanks@jpl.nasa.gov or William.H.Duquette@jpl.nasa.gov. Note that this document is also available in hardcopy.

Software Manual Pages

Extensive documentation of the Mars software tools and libraries is included in the software installation set in the form of software “man pages”.

2. MARS CONCEPTS

This section gives an overview of Mars and the concepts shared by its various models. The discussion is kept to a high level; see Sections 3 and following for detailed models.

2.1 The Client Simulation

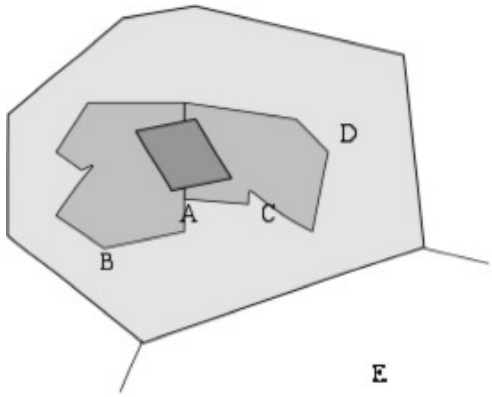
Mars is an infrastructure library; its models are intended for use in other simulation applications. These are referred to as *client simulations*, or simply as *clients*.

2.2 GRAM: Modeling the Population

GRAM, the Generalized Regional Attitude Model, models the attitudes of the civilian groups that reside in a particular region, and of the organization groups that work in the region. GRAM is described in detail in Section 3.

2.3 The Playbox

JNEM models population dynamics in a geographical region called the *playbox*. The playbox is divided into areas called *neighborhoods*. Neighborhoods are simply a way of dividing the playbox into a number of reasonably homogeneous areas, and may be of any size: country, province, city, town, zip code, and neighborhood proper. Geographically, neighborhoods are defined as polygons whose vertices are defined by latitude/longitude coordinates. Neighborhoods may nest, e.g., a city may be a "neighborhood" within a larger province, and the city may contain several neighborhoods. In the diagram below, "A" is an urban area surrounded by suburban areas B and C; all three lie within D, a county, which abuts E, another county.



Note that locations within an inner neighborhood are not also part of the outer neighborhood; in the diagram above, D effectively has a hole cut out of it by B and C, and A cuts sections out of both B and C[1]. Consequently, if a neighborhood is completely tiled by nested neighborhoods, it can be omitted as it contains no locations. If D's surface were entirely covered by suburbs, for example, there would be nothing left of D and no reason to define it.

In GRAM, simulation events take place within neighborhoods, and affect the population of the neighborhoods. The geographic spread of the ripple effects of an event taking place in a neighborhood depends on how nearby other neighborhoods are presumed to be--not simply geographically but also socially. The nearness of one neighborhood to another is an input to GRAM called *neighborhood proximity*. There are four proximity levels: *here*, *near*, *far*, and *remote*. The diagram above shows proximity to neighborhood A. From A's point of view, A is here. Suburbs B and C are *near* A, and outlying area D is *far* from A. Neighborhood E is *remote*.[2] An event in A would affect A immediately, would likely affect B and C, though to a lesser degree, might affect D to a much lesser degree, and would not affect E at all. Ripple effects in other neighborhoods can also be delayed by an interval, which is an input for each pair of neighborhoods.

2.4 Population Groups

The people in the playbox are divided into *population groups*, of which there are three kinds: civilian groups, organization groups, and force groups.

2.4.1 Civilian Groups

Civilian groups represent the population of the playbox, i.e., the people who actually live in the neighborhoods. This population maybe broken into groups by ethnicity, religion, language, social class, political affiliation, or any other demographic criteria the analyst deems necessary. Groups are similar to the "market segments" used to target advertising: a group is a collection of people who may be assumed to have similar biases, interests, and behaviors due to their demographic similarity.

GRAM tracks the civilian population by group and neighborhood; each neighborhood must have a non-zero population of at least one of the civilian groups, and may include representatives of all of the civilian groups. A civilian group in a specific neighborhood is often referred to as a *neighborhood group*.

In addition to each civilian group's population in each neighborhood, each civilian group might also be represented in the client simulation in the form of civilian units. People present in units are referred to as *personnel*, rather than as *population*.

---

1 Although GRAM allows neighborhoods that overlap without nesting, like A, other applications may require that overlapping neighborhoods be properly nested. In practice, consequently, the borders of B and C must go around A, rather than A being overlaid on top of them.

2 Note that these proximities are social, not geographic—proximities are input to GRAM, and are not computed from the geometry of the neighborhoods or the distance from one neighborhood to another.

GRAM models neighborhood groups in detail, tracking the attitudes of each group in each neighborhood along several axes as the group's members are affected by events and situations taking place in the client simulation. These attitudes then will typically affect the group's reactions and responses in the client simulation.

2.4.2 Organization Groups

Organization groups represent organizations that are present in the playbox to help the civilians. There are three kinds: Non-Governmental Organizations (NGOs), International or Intergovernmental Organizations (IGOs), and Contractors (CTRs). NGOs are groups like the Red Cross or Doctors Without Borders who do humanitarian relief, development, and so forth. IGOs are international organizations like UNESCO. Contractors are commercial firms who are doing development work in the playbox, often but not necessarily working for the Coalition. Organizations may be either local or foreign.

Organization group members are represented in the federation as ground units; organization units may conduct a variety of activities which affect civilian satisfaction. Each organization group may have medical, engineer, and/or support capabilities; the activities a group's units may perform depend on the group's capabilities.

GRAM tracks the attitude of each organization group toward working in each neighborhood; within the client simulation, a group that is willing to work in a low-risk neighborhood might be unwilling to work in a neighborhood it perceives as high-risk.

2.4.3 Force Groups

Force groups represent military forces, such as the U.S. Army, and other groups whose purpose is to apply force. United States forces are generally referred to as "BLUE", and so by convention belong to the "BLUE" force group. There are five kinds of force group:

- Regular military, e.g., BLUE
- Paramilitary, e.g., SWAT teams and other combat-trained police units
- Police, e.g., normal civilian police
- Irregular military, e.g., militias
- Criminal, e.g., organized crime

None of the models presently in Mars deal with force groups, but client simulations generally do.

2.5 Simulated Time

Mars measures simulated time in integer *ticks*. The duration of one tick can be anything from one second to two minutes to three hours to four or more days; tick sizes of one minute and of one day are typical. The simclock(n) module tracks simulated time, and converts between ticks and hours, minutes, and seconds; it also supports military "Zulu-time" strings.

In addition, GRAM expresses rates of attitudinal change as points per decimal day; it automatically converts between decimal days and ticks.

Client simulations will often have a minor time step, the tick, and also one or more major time steps; these are accordingly called *tocks*. In JNEM, for example, the tick is one minute, and GRAM is advanced at the tock, once every five ticks.

3. GENERALIZED REGIONAL ATTITUDE MODEL

The Generalized Regional Attitude Model (GRAM) is a population dynamics model of the attitudes and behavior of groups within neighborhoods within the playbox. GRAM tracks changes in attitudes over time. Changes are driven by events and situations modeled within the client simulation (e.g., civilian casualties, presence of force units in a neighborhood, and so forth). The client simulation will usually use rule sets to analyze these drivers and provide attitude inputs to GRAM.

The effects of an attitude driver are not limited to the neighborhood and group that are directly affected by the driver. There are second order effects on other groups in the neighborhood, and on groups in other neighborhoods. These *indirect effects* generally weaken with distance; they can also be delayed in time.

As simulation time progresses, GRAM tracks the contribution of each driver to each attitude curve, thus enabling the significant drivers to be determined after the fact.

At present, GRAM supports two different kinds of attitude: the satisfaction of civilian and organization groups with respect to various concerns, and the cooperation of civilian groups with force groups.

This section details the nature of attitude levels and curves, the kinds of inputs that can affect each, the spread of indirect effects across the playbox, and how the results are computed. The specifics of satisfaction and cooperation levels are described in detail.

GRAM is a generalization and extension of the JNEM Regional Analysis Model (JRAM), as developed for the Joint Non-kinetic Effects Model (JNEM), which was in turn based on an earlier model called the Regional Analysis Model, or RAM. RAM was developed for the National Simulation Center by the Texas A&M University's Department of Political Science, working with the George Bush School of Government and Public Service and the Texas Center for Applied Technology (both also at Texas A&M). RAM was part of the Spectrum Simulation to model biases, alliances, rivalries, and other aspects of inter-group relationships.

3.1 Attitude Curves

A group's satisfaction and cooperation levels are collectively referred to as the group's *attitudes*. Attitudes will change over time, depending on what happens to the group. Events and situations that affect a group's attitudes are called *attitude drivers*, or simply *drivers*. As an attitude changes over time, it traces out an *attitude curve*, $A(t)$; the value of the attitude at time 0 is denoted $A(0)$. The chosen unit of time is irrelevant to the model, but is usually decimal days. The value of $A(t)$ may range from A_{min} to A_{max} (–100.0 to +100.0 for satisfaction curves, and 0.0 to 100.0 for cooperation curves).

More specifically, the satisfaction for group g in neighborhood n with respect to concern c at time t is denoted $S_{ngc}(t)$, and the cooperation for neighborhood group nf with force group g at time t is denoted

$\Omega_{nfg}(t)$. The *initial satisfaction* is $S_{ngc}(0)$ and the *initial cooperation* is $\Omega_{nfg}(0)$.

Any attitude curve $A(t)$ is recomputed at a series of major time steps (tocks) $t_1, t_2, t_3, \ldots$, as follows:

$$\begin{aligned} A(t_1) &= A(0) + (\text{contributions from 0 to } t_1) \\ A(t_2) &= A(t_0) + (\text{contributions from } t_1 \text{ to } t_2) \\ &\vdots \end{aligned}$$

For simplicity in notation during the following discussion, we will use t_1 to denote the time of the current tock and t_0 to denote the time of the previous tock. The interval $(t_1 - t_0)$ is typically constant for all pair of tocks, but in principle the interval could vary from tock to tock. The "contributions" at each tock are due to attitude drivers and are of three kinds:

- The effect of the long-term trend (for satisfaction curves only; see Section 3.3.4 .
- The contribution of level effects
- The contribution of slope effects

JIN's rule sets analyze current drivers, producing level inputs and slope inputs on particular attitude curves. GRAM translates these inputs into level effects and slope effects as described in Section 3.5 ; these effects contribute to the attitude curves as described in Sections 3.1.1 and 3.1.2 respectively.

3.1.1 Level Effects

GRAM inputs can create *level effects* on attitude curves. A level effect is a nominal change of a specified magnitude which takes place over a specified period of time, typically 0.1 days. The rate at which the effect's magnitude, or *limit*, is realized is defined by a *realization curve*. This section explains how to compute the nominal contribution of a particular level effect to an attitude curve during some time step.

3.1.1.1 Level Effects Defined

A level effect can be thought of as a tuple of the following elements:

- The indices of the affected attitude curve (*ngc* for satisfaction curves, *nfg* for cooperation curves)
- *d*, the ID of the driver that resulted in this affect.
- *k*, an indicator of the effect's cause; see Section 3.1.5 .
- *limit*, the nominal magnitude of the change

- *days*, the time interval in days over which the effect is realized
- *ts*, the effect's start time
- *te*, the effect's end time.
- τ, a parameter which controls the shape of the realization curve.

When working with a set of level effects, these parameters can be subscripted with the effect index *i*, e.g. $limit_i$ is the *limit* of the *i*th effect.

The realization curve for a level effect is defined by the following function $E(t)$:

$$E(t) = \begin{cases} 0 & t \le ts \\ limit \cdot \left(1.0 - e^{\frac{-(t-ts)}{\tau}}\right) & ts < t < te \\ limit & t \ge te \end{cases}$$

This exponential curve approaches but never actually reaches the *limit*. Consequently, τ is chosen that the curve is just ϵ from *limit* at time *te*:

$$\tau = \frac{days}{-\ln\left(\frac{\epsilon}{|limit|}\right)}$$

$E_i(t)$ denotes this function for a specific level effect *i*. The nominal contribution of level effect *i* at time step t_1 is therefore

$$\delta_i(t_1, t_0) = E_i(t_1) - E_i(t_0)$$

Note that those level effects with $te < t_0$ or $ts > t_1$ are guaranteed to contribute a value of 0 to the curve during the time step—they have either run their course before the interval starts, or have not begun to have an effect when the interval ends. For such effects,

$$\delta_i(t_1, t_0) = 0$$

This nominal contribution $\delta_i(t_1, t_0)$ is used to compute the actual contribution of effect *i* during the time step. In addition, as the simulation runs from time step to time step we accumulate the nominal contribution to date for each level effect *i*:

$$ncontrib_i(t_1) = ncontrib_i(t_0) + \delta_i(t_1, t_0)$$

Accumulating the nominal contribution to date for a level effect is useful because it allows us to watch the progress of the level effect in terms of the original inputs as the model runs forward in time.

3.1.1.2 Effect of Epsilon on Level Effects

GRAM uses an ϵ value[3], nominally equal to 0.1, that effects level effects in two ways:

- As described in Section 3.1.1.1 , ϵ is used to calibrate the exponential curve so that it reaches ϵ of its *limit* at its end time, *te*.
- When scheduling level effects the requested realization time in *days* is ignored for effects whose $limit < \epsilon$. Instead, *te* is set to *ts* and *days* is set to 0, so that the effect makes its entire contribution at time step containing *ts*.

3.1.2 Slope Effects

GRAM inputs can create *slope effects*. A slope effect is an attitude change with a specified nominal slope (change/day). The effect will cause the attitude to change at that same nominal rate until its end time has been reached, or indefinitely if no end time is specified. This section explains how to compute the nominal contribution of a particular slope effect during some time step.

3.1.2.1 Slope Effects Defined

A slope effect can be thought of us a tuple of the following elements:

- The indices of the affected attitude curve (*ngc* for satisfaction curves, *nfg* for cooperation curves).
- *d*, the ID of the driver that resulted in this effect
- *k*, an indicator of the effect's cause; see Section 3.1.5
- *slope*, the nominal change per day
- *ts*, the effect's start time in ticks
- *te*, the effect's end time in ticks

When working with a set of slope effects, these parameters can be subscripted with the effect index *i*, e.g., $slope_i$ is the *slope* of the *i*th effect.

3 Defined in the model parameter database as `jram.epsilon`.

It would seem that the nominal contribution of a slope effect *i* for time step t_1 is simply the *slope* times the duration of the time step:

$$\delta_i(t_1, t_0) = slope \cdot (t_1 - t_0)$$

However, the effect might not apply for the full time interval; indeed it might not apply for any of the time interval. There are the following cases:

- $ts \geq t_1$, in which case the effect hasn't yet started to contribute to satisfaction.
- $te \leq t_0$, in which case the effect is no longer contributing to satisfaction.
- $ts < t_1$ and $te > t_0$, in which case the effect is contributing for some or all of the interval.

Consequently, we define

$$\delta_i(t_1, t_0) = \begin{cases} slope \cdot (\min(te, t_1) - \max(ts, t_0)) & \text{where } ts < t_1 \text{ and } te > t_0 \\ 0 & \text{otherwise} \end{cases}$$

As the simulation runs from time step to time step we accumulate the nominal contribution to date for each slope effect, just as we did for level effects:

$$ncontrib_i(t_1) = ncontrib_i(t_0) + \delta_i(t_1, t_0)$$

However, we must also consider *slope chains*.

3.1.2.2 Situations and Slope Chains

A slope chain is a sequence of slope effects related to a single driver, usually a *situation*[4], that target the same attitude curve. Chains are produced when a situation evolves over time, producing a sequence of slopes. The essential thing about effects in a slope chain is that they may not overlap in time, precisely because the chain really represents a single effect that happens to fluctuate over time.[5] (Section 3.5 describes the genesis of slope chains in detail).

The following table represents a slope chain for some particular situation *d*. Note that all effects in the chain have the same *d*, curve indices, and *k*; these values are therefore omitted from the table:

4 A *situation* is an on-going condition, known to the client simulation, that has satisfaction implications for as long as it lasts.
5 And, in fact, it's implemented as a single effect, which contains a list of future start times and slopes.

TS	TE	SLOPE
7	24	5.2
24	39	0
39	108	2.5
108		7.1

The situation begins at tick 7, and the slope is 5.2. At tick 24 the situation becomes inactive, and the slope drops to 0. At tick 39 the situation becomes active again, with a slope of 2.5. At tick 108 things heat up and the slope rises to 7.1. Note that the final effect in the chain has no end time; slope chains are terminated by setting the slope to 0 in the final link in the chain.

The presence of slope chains affects the model in two ways. First, care must be taken when scheduling a new effect to update any exist chain. If the situation above were to change again at time 134, for example, the existing chain would need to be extended with the new slope.

Second, it's possible that several links in a chain will contribute to satisfaction during a single time step, e.g., if the slope changes several times during the time step. Since the links really represent a single fluctuating effect, the total nominal contribution of the links must be considered when computing the actual contribution of the chain.

Extending the equations described in the previous sections to account for slope chains would be a tedious exercise in notation. To summarize, then, the implementation must account for the following:

- A new link in a slope chain must terminate any previous link in that chain.
- The nominal contribution must be computed for the chain as a whole.
- If two or more links in a chain are active during a single time step, they must be treated as a single effect with respect to the computation of the actual contribution to the attitude curve.

3.1.2.3 Effects of Epsilon on Slope Effects

The same ϵ used in the scheduling and assessment of level effects is applied to slope effects in a different way. If, when a slope effect is being scheduled, its $slope < \epsilon$, arbitrarily treated as though its $slope = 0$. This prevents the computation of insignifcant slope effects (and particularly of insignificant indirect effects) from affecting performance.

3.1.3 Ascending and Descending Thresholds

Every level and slope effect has, in addition to the elements shown above, an *ascending threshold* and a

descending threshold, which are used to filter out effects when the nominal contribution is computed. These thresholds are denoted *athresh* and *dthresh*. The notion is that the given effect cannot increase the underlying curve *A* if $A \geq athresh$, and cannot decrease the underlying curve *A* if $A \leq dthresh$. Thus, a given satisfaction slope effect might have a slope of -5.0 and a *dthresh* of -20: the curve will lose 5 nominal points per day, but only up to a threshold of -20 satisfaction points. The effect is reasonably sharp, but cannot make the group more than mildly annoyed. On the other hand, so long as the effect is in place it may be difficult to improve the group's mood much above -20.

The handling of thresholds is straightforward: effects for which the underlying curve exceeds the relevant threshold are treated as though their nominal contribution were zero.

The thresholds are set when the effect is scheduled; see Section 3.5.7 .

A judicious use of thresholds can greatly enrich the notion of the long-term-trend; see Section 3.3.4 .

3.1.4 Scaling of Contributions

The actual contribution to any attitude curve $A(t)$ should show the effects of diminishing returns (technically, *diminishing marginal utility*) as the extreme values are approached. Specifically:

- Positive contributions should have a stronger effect when $A(t)$ is near A_{min} and a weaker effect when $A(t)$ is near A_{max}.
- Negative contributions should have a stronger effect when $A(t)$ is near A_{max} and a weaker effect when $A(t)$ is near A_{min}.
- $A(t)$ should stay within the range A_{min} to A_{max} without being artificially clamped.

To achieve this, each non-zero *nominal contribution* to $A(t_1)$ is scaled given the value of $A(t_0)$, producing the *actual contribution*. The following scheme has the desired properties, provided that the total actual nominal contributions at each time step are less than $A_{max} - A_{min}$.

First, for each nominal contribution *ncontrib* to satisfaction curve $S(t_0)$, let

$$\text{scale}(ncontrib) = \begin{cases} 2 \cdot \dfrac{100 - S(t_0)}{200} \cdot ncontrib & \text{where } ncontrib \geq 0 \\ 2 \cdot \dfrac{100 + S(t_0)}{200} \cdot ncontrib & \text{where } ncontrib < 0 \end{cases}$$

With this formula, a nominal contribution of 10 points will cause the satisfaction level to move 10% of the distance from its current value toward the upper limit, +100, no matter what that current value actually is. Similarly, a nominal contribution of −10 points will cause the satisfaction level to move 10% of the distance from its current value toward the lower limit, −100. Consequently, nominal contributions can be thought of either as points *or* as percentage changes in the difference between the current value and the extreme.

Similarly, for each nominal contribution *ncontrib* to cooperation curve $\Omega(t_0)$, let

$$\text{scale}(ncontrib) = \begin{cases} \dfrac{100 - \Omega(t_0)}{100} \cdot ncontrib & \text{where } ncontrib \geq 0 \\ \dfrac{\Omega(t_0)}{100} \cdot ncontrib & \text{where } ncontrib < 0 \end{cases}$$

In the sections that follow, the notation "$\text{scale}(x)$" indicates that *x* has been scaled in this way.

3.1.5 Causes and Scaling

The discussions in Sections 3.1.1.1 and 3.1.2.1 show how to compute the nominal contribution of each level and slope effect during a given time step. Section 3.1.4 shows how the contributions can be scaled so that the attitude level does not get clamped at its maximum or minimum value. There is one more piece to the puzzle.

We could simply compute the actual contribution during the time step from t_0 to t_1 as follows:

$$acontrib(t_1) = \text{scale}\left(\sum_i \delta_i(t_{1,} t_0)\right)$$

This, however, presumes that the collection of level and slope effects active at a given time are all independent of one another, and that each should always contribute its full scaled magnitude. This is not necessarily the case. Level and slope effects are the result of independent drivers which affect the local civilian population and hence affect their attitudes. But even if the drivers are independent, the effects need not be.

People's capacity to respond to events and situations, their ability to feel horror and dismay on the one hand or joy and exultation on the other, can be saturated on a number of axes. Once their capacity is saturated due to drivers of a particular kind, further events of that kind occuring shortly thereafter are unlikely to have much additional effect. Consider, for example, a neighborhood which is experiencing a serious epidemic. It's unlikely that a second epidemic afflicting the neighborhood at the same time will change the results much.

GRAM handles this through the notion of *causes*. Each input to GRAM can be assigned a *cause*. Inputs due to similar drivers—e.g., bombings—will have identical causes. All effects stemming from a single input will share that input's cause.

When contributions to a single curve $A(t)$ share a single cause, their total contribution is the contribution of the largest effect. More precisely, for each cause *k* let I_k^+ be the set of effects *i* that have cause *k* and for which $\delta_i(t_1, t_0) > 0$; similarly, let I_k^- be the set of effects *i* that have cause *k* and for which $\delta_i(t_1, t_0) < 0$. The nominal contribution of the effects with cause *k* is then

$$\max_{i \in I_k^+} \delta_i(t_1, t_0) + \min_{i \in I_k^-} \delta_i(t_1, t_0)$$

If we treat GRAM inputs for which no cause is specified as having a unique cause *k*, then the attitude level at time step t_1 is

$$A(t_1) = A(t_0) + \text{scale}\left(\sum_k \max_{i \in I_k^+} \delta_i(t_1, t_0) + \min_{i \in I_k^-} \delta_i(t_1, t_0) \right)$$

We would then like to compute the actual contribution of each effect *i* to this final result. The scaled contribution of each level effect *i* with cause *k* is shared with the other effects with cause *k* that are active in that time step. The following equation allocates the total scaled contribution back to each of the constituent effects on a *pro rata* basis, resulting in the effect's *actual contribution to date*:

$$acontrib_i(t_1) = acontrib_i(t_0) + \begin{cases} \text{scale}\left(\dfrac{\delta_i(t_1, t_0)}{\sum_{j \in I_k^+} \delta_j(t_1, t_0)} \cdot \max_{j \in I_k^+} \delta_j(t_1, t_0) \right) & \text{if } i \in I_k^+ \\ \text{scale}\left(\dfrac{\delta_i(t_1, t_0)}{\sum_{j \in I_k^-} \delta_j(t_1, t_0)} \cdot \min_{j \in I_k^-} \delta_j(t_1, t_0) \right) & \text{if } i \in I_k^- \end{cases}$$

Accumulating the actual contributions to date is useful because it allows us to see precisely how the effect has changed $A(t)$—and ultimately, how a given driver has changed attitudes in general.

3.2 Neighborhoods and Groups

Having discussed attitude curves in general, it's now time to discuss satisfaction and cooperation curves in specific. But first, we must discuss neighborhoods and groups.

GRAM models attitudes in a geographical region called the *playbox*. The playbox is divided into N_n areas, which are called *neighborhoods*. Similarly, the civilian population of the playbox is divided into N_g groups. A group *g* is said to reside in neighborhood *n* if $population_{ng} > 0$, that is, if the group has non-zero population in the neighborhood. Note that civilian group *g* resident in neighborhood *n* is sometimes referred to as neighborhood *ng*.

GRAM tracks satisfaction for each neighborhood group, and also tracks cooperation for each neighborhood group with each force group.

Organization groups (i.e., humanitarian relief groups) are assumed to have their identity playbox-wide, and do not reside in neighborhoods as such. Nevertheless, organization groups work in neighborhoods, and consequently have attitudes about each neighborhood. GRAM therefore tracks each organization group's satisfaction within each neighborhood.

3.3 Satisfaction

GRAM tracks the satisfaction of civilian and organization groups, with respect to each of the groups' concerns, in neighborhoods. Summary statistics are computed for neighborhoods and for the playbox as a whole.

3.3.1 Satisfaction Levels

The degree to which a group is satisfied with its condition with respect to some concern is described by a satisfaction value, sometimes called a *satisfaction level*. A satisfaction value is a decimal number *S*, where $-100.0 \le S \le +100.0$. A value of +100 denotes perfect satisfaction, and a value of -100.0 denotes utter dissatisfaction. The following rating scale is frequently used:[6]

SYMBOL	MEANING	MIDPOINT	RANGE
VS	Very Satisfied	80.0	$60.0 \le S \le 100.0$
S	Satisfied	40.0	$20.0 \le S < 60.0$
A	Ambivalent	0.0	$-20.0 \le S < 20.0$
D	Dissatisfied	-40.0	$-60.0 \le S < -20.0$
VD	Very Dissatisfied	-80.0	$-100.0 \le S < -60.0$

Note that this is a relative scale. Satisfaction is measured on several axes, and different groups place different weights on different axes. Thus, satisfaction values must generally be weighted in order to be comparable. See the discussion of saliency in Section 3.3.3 .

6 This scale is implemented by the simlib(n) type `qsat`.

3.3.2 Concerns

A group may be satisfied or dissatisfied along several different axes, e.g., personal safety, quality of life, and so forth. In GRAM these axes are called *concerns*. GRAM tracks satisfaction with respect to concerns for groups within neighborhoods.

GRAM itself doesn't care what the concerns are; the client simulation is free to define any concerns it chooses. However, there are standard civilian and organization concerns that have generally been used over the last several years. The set of civilian concerns is as follows:

Autonomy (AUT): Does the group feel it can maintain order and govern itself with a stable government and a viable economy?

Safety (SFT): Do members of the group fear for their lives, either from hostile attack or from collateral damage from force activities? This fear includes environmental concerns such as life-threatening disease, starvation, and dying of thirst.

Culture (CUL): Does the group feel that its culture and religion, including cultural and religious sites and artifacts, are respected or denigrated?

Quality of Life (QOL): QOL includes the physical plants that provide services, including water, power, public transportation, commercial markets, hospitals, etc., and those things associated with these services, such as sanitation, health, education, employment, food, clothing, and shelter.

The standard set of organization concerns is as follows:

Casualties (CAS): How dangerous the ORG group views its environment to be in relationship to its members' willingness to risk (or continue risking) their lives to do their work. If CAS is sufficiently low, members of the group might refuse to work.

Service (SVC): The group's satisfaction with the service it is providing to the civilians. ORG groups have higher SVC satisfaction when performing good works and lower satisfaction when inactive.

3.3.3 Composite Satisfaction, Weights, and Saliencies

The satisfaction of group *g* in neighborhood *n* with respect to concern *c* is denoted S_{ngc}. It is frequently desirable to summarize the full S_{ngc} matrix using a variety of weighted averages, collectively termed *composite satisfaction* values. For example, we are often interested in group *g*'s composite satisfaction in neighborhood *n*; this is called the neighborhood group's *mood*, and is denoted S_{ng}.

We could compute such composites using a simple average, but that would neglect two important factors. First of all, a group usually places more importance on some concerns than others, and

different groups often place importance on different concerns. The importance[7] a group places on a concern is called the group's *saliency* for the concern. Saliency is represented by a number L_{ngc} where $0.0 \le L_{ngc} \le 1.0$. The following rating scale is used:

SYMBOL	MEANING	VALUE
CR	Crucial	1.00
VI	Very Important	0.85
I	Important	0.70
LI	Less Important	0.55
UN	Unimportant	0.40
NG	Negligible	0.00

When averaging across concerns we must weight by the saliency of each. Note that this is an absolute scale.

Second, different groups are of different sizes, and some groups have more importance to the wider community than other groups. We represent this by defining a GRAM input called the *rollup weight* of group *g* in neighborhood *n*, denoted by W_{ng}. This weight is nominally 1.0, and will typically have a value between 0.5 and 2.0. When averaging across groups or neighborhoods we must weight by the rollup weight. Note that W_{ng} will usually be correlated with $population_{ng}$, in that the size of a neighborhood group will affect its importance to the group as a whole; however, a small elite group can have a higher weight than a large underclass.

It has been shown[8] that the following equation properly "rolls up" satisfaction across any set *A* of neighborhoods, groups, and concerns:

$$S_A = \frac{\sum_A W_{ng} \cdot L_{ngc} \cdot S_{ngc}}{\sum_A W_{ng} \cdot L_{ngc}}$$

Given this, we can define the following useful composite satisfactions:

The mood of each group *g* in each neighborhood *n*:

7 Gurr defines *salience* as "the strength of motivation to attain or maintain the desired value position" in *Why Men Rebel*, Ted Gurr, Princeton, NJ: Princeton University Press. 1970, p. 66. This book was a seminal source for the original RAM model.

8 "Satisfaction Roll-Up", Robert G. Chamberlain, September 12, 2006

$$S_{ng} = \frac{\sum_{c} W_{ng} \cdot L_{ngc} \cdot S_{ngc}}{\sum_{c} W_{ng} \cdot L_{ngc}}$$

The top level or *playbox satisfaction* of each group *g* with each concern *c*:

$$S_{gc} = \frac{\sum_{n} W_{ng} \cdot L_{ngc} \cdot S_{ngc}}{\sum_{n} W_{ng} \cdot L_{ngc}}$$

The top level of *playbox mood* of each group *g*:

$$S_{g} = \frac{\sum_{n} \sum_{c} W_{ng} \cdot L_{ngc} \cdot S_{ngc}}{\sum_{n} \sum_{c} W_{ng} \cdot L_{ngc}}$$

3.3.4 Long-Term Trend

The *long-term trend*, $\sigma_{ngc}(0)$, is a constant that produces a steady increase of decrease in a satisfaction curve over time. It is expressed as a percentage change in the satisfaction level per day. As shown, it may vary by neighborhood, group, and concern. In future versions of GRAM the long-term trend might vary over time, hence we denote it as an initial condition at time 0.

The long-term trend is implemented as a slope effect on each satisfaction curve S_{ngc}, for a pseudo-driver called "TREND". This pseudo-driver has a direct effect on each curve, with nominal magnitude $\sigma_{ngc}(0)$; there are no indirect effects.

The use of thresholds (Section 3.1.3) can greatly enrich the use of long-term trends. To begin with, any slope effect, if applied over a long period of time, will drive satisfaction to one of the extremes (-100.0, +100.0). Thresholds limit long-term trends to a more reasonable effect.

Long-term trends can be even more useful when implemented in pairs, $sigma^{+}_{ngc}(0)$ and $sigma^{-}_{ngc}(0)$. For example, regression to a mean (0.0, say) can be implemented by creating a positive slope effect with an *athresh* of 0.0 and a negative slope effect with a *dthresh* of 0.0. When the curve is negative, the first will drive it up to 0.0; when it is negative, the second will drive it down to 0.0.

See Section 3.1 for more information about slope effects, and Section 3.5 for more information about drivers.

3.4 Cooperation

GRAM tracks the *cooperation* of neighborhood groups with force groups. Cooperation is a concept that derives from the TRADOC HUMINT methodology; it indicates the likelihood that one group *f* will provide intelligence to another group *g*. Note that cooperation is distinct from providing active aid to another group; this is sometimes called *collaboration*. GRAM does not model collaboration at this time.[9]

3.4.1 Cooperation Levels

The probability that a member of neighborhood group *nf* will provide intelligence to a member of force group *g* is expressed as a number between 0 and 100. The *cooperation level* between *nf* and *g* is denoted $\Omega_{nfg.}$. Note that the information flow is always from group *nf* to group *g*, i.e., from the first group to the second.

Cooperation levels are often represented symbolically, as indicated in the following table:

SYMBOL	MEANING	VALUE	RANGE
AC	Always Cooperative	100.0	$99.9 \le \Omega \le 100.0$
VC	Very Cooperative	90.0	$80.0 \le \Omega < 99.9$
C	Cooperative	70.0	$60.0 \le \Omega < 80.0$
MC	Marginally Cooperative	50.0	$40.0 \le \Omega < 60.0$
U	Uncooperative	30.0	$20.0 \le \Omega < 40.0$
VU	Very Uncooperative	10.0	$1.0 \le \Omega < 20.0$
NC	Never Cooperative	0.0	$0.0 \le \Omega < 1.0$

3.4.2 Composite Cooperation

The cooperation of neighborhood *n* as a whole with force group *g* is denoted Ω_{ng}, and is computed as follows:

$$\Omega_{ng} = \frac{\sum_{f} population_{nf} \times \Omega_{nfg}}{\sum_{f} population_{nf}}$$

9 Ambassador Terry McNamara has pointed out that "collaboration" is a loaded word, and that a more neutral term is preferable.

Note that we weight cooperation by the size of the population, rather than by more abstract weights and saliencies, as we do when computing composite satisfactions. The cooperation of group *f* with group *g* can be thought of as the probability that an arbitrary member of group *f* will give intel to group *g*. Thus, when averaging across groups it makes sense to weight by the number of members in each.

3.5 Drivers, Inputs, and Effects

The client simulation analyzes attitude drivers (events and situations) and produces satisfaction and cooperation inputs for GRAM. This section explains how each input is turned into a collection of effects.

Each GRAM input specifies a direct effect on a particular attitude curve. This direct effect is a level or slope effect as described in Sections 3.1.1 and 3.1.2 . The direct effect then engenders indirect effects on other attitude curves.

Each input optionally includes an ascending threshold (*athresh*) and descending threshold (*dthresh*) as described in Section 3.1.3 . If not specified, these thresholds default to the minimum and maximum of the curve's range, i.e., the default *athresh* and *dthresh* for satisfaction inputs are +100.0 and -100.0. Each effect engendered by the input inherits the input's thresholds. This, incidentally, is why both thresholds are required. An *athresh* is only binding on a positive effect--but a positive input can produce both positive and negative effects.

Indirect effects differ from the direct effect that engenders them in two ways. First, their magnitude (*limit* for level effects, *slope* for slope effects) is affected by the relationship between the group directly effected and the group indirectly effected, and by the relationship between the neighborhoods in which they reside. Second, indirect effects in other neighborhoods are delayed by an interval that depends on the two neighborhoods.

This section explains how to determine the magnitude and delay for each indirect effect. Before we can define them, however, we must first define the proximity and effects delay between two neighborhoods.

3.5.1 Neighborhood Proximities

Whether or not a GRAM input in neighborhood *n* has indirect effects in another neighborhood *m* depends in part of the proximity of neighborhood *m* to neighborhood *n*, and on the nature of the driver. For example, combat in a neighborhood will likely concern residents in nearby neighborhoods, and might even concern residents in far away neighborhoods. Other situations, such as a sewage spill, might only concern residents of the neighborhood in which the spill occurs.

Neighborhood proximity is defined by the $proximity_{mn}$ matrix. It defines the proximity of neighborhood *n* to neighborhood *m* from the point of view of residents of *m*. Each element of the matrix must have one of the following values:

VALUE	SYMBOL	NOTES
0	HERE	*m* = *n*; indirect effects always occur.
1	NEAR	*n* is near *m*; indirect effects are common.
2	FAR	*n* is far from *m*; indirect effects are rare.
3	REMOTE	*n* is remote from *m*; indirect effects never occur.

Note that proximity, in this sense, is not simply a measure of miles walked or driven, but rather a perception on the part of the residents of neighborhood *m*. For example, neighborhoods A and B might be adjacent on the map, but have a natural boundary (e.g., a river) between them, such that residents of A seldom travel to B; thus, residents of A might consider B to be FAR. Neighborhood C might be farther from A than B is, but C contains a popular destination, e.g., a shopping district, or an industrial district that employs many of the residenta of A. The residents of A would then consider C to be NEAR, even thought it's farther away than B, which is considered to be FAR.

Also, note that $proximity_{mn}$ need not be symmetric, that is, it is okay if $proximity_{mn} \neq proximity_{nm}$. In our example, the residents of A frequently visit C and so regard C as NEAR; but if the residents of C seldom visit A, they might regard A as FAR.

For convenience, we also define the following symbols:

$$here_{mn} = \begin{cases} 1 & \text{if } proximity_{mn} = 0 \\ 0 & \text{if } proximity_{mn} \neq 0 \end{cases}$$

$$near_{mn} = \begin{cases} 1 & \text{if } proximity_{mn} = 1 \\ 0 & \text{if } proximity_{mn} \neq 1 \end{cases}$$

$$far_{mn} = \begin{cases} 1 & \text{if } proximity_{mn} = 2 \\ 0 & \text{if } proximity_{mn} \neq 2 \end{cases}$$

$$remote_{mn} = \begin{cases} 1 & \text{if } proximity_{mn} = 3 \\ 0 & \text{if } proximity_{mn} \neq 3 \end{cases}$$

3.5.2 Proximity Limits

Because a single GRAM input can produce a vast quantity of indirect effects, performance may become an issue in practice. GRAM implements the notion of the *proximity limit*[10] as a means to cope with GRAM performance issues during training exercises. The proximity limit may be set as indicated in the following table:

10 The proximity limit is found in the model parameter database as "`jram.proxlimit`".

LIMIT	CONSEQUENCES
far	An input in neighborhood *n* may have indirect effects in *n*, in neighborhoods near *n*, and in neighborhoods far from *n*.
near	An input in neighborhood *n* may have indirect effects in *n* and neighborhoods near *n*. If the proximity limit is changed to "near", existing indirect effects in far neighborhoods will be terminated.
here	An input in neighborhood *n* will have indirect effects only in *n*. If the proximity limit is changed to "here", existing indirect effects in near and far neighborhoods will be terminated.
none	No indirect effects will be scheduled. If the proximity limit is changed to "none", all existing indirect effects will be terminated.

3.5.3 Neighborhood Effects Delay

When an event occurs in neighborhood *n*, the response to that event in another neighborhood *m* will often be delayed: news takes time to spread. This delay is probably correlated with the distance between the two neighborhoods, but as with proximity geographic distance doesn't tell the whole story. Given cell phones and the Internet, communications between any two points can be nearly instantaneous, making distance moot; and yet, just because news *can* travel instantly doesn't mean that it does. Thus we define $delay_{mn}$ as the time delay in decimal days for an input in neighborhood *n* to have an indirect effect in neighborhood *m*. Note that $delay_{mn} = 0$ *for all* n*;* there is no delay within neighborhood *n* itself.

3.5.4 Satisfaction Influence

As stated above, a direct effect on a particular neighborhood group's concern can have indirect effects on the same concern[11] for other groups in the same and other neighborhoods. The range of possible indirect effects is called the neighborhood's *satisfaction influence*; it depends upon the relationships between the group's neighborhood and other neighborhoods, upon the relationships between the group and other groups, and upon the group's importance. In particular, the satisfaction influence si_{ngmf} of group *ng* on group *mf* is a multiplicative factor that is computed as follows:

11 This implies that direct effects on civilian groups have no effect on the satisfaction of organization groups, and *vice versa*, because civilian groups and organization groups have no shared concerns.

$$si_{ngmf} = \begin{cases} rel_{mfg} & proximity_{mn} = 0 \\ rel_{mfg} \cdot effects\_factor_{ng} & proximity_{mn} = 1 \\ rel_{mfg} \cdot effects\_factor_{ng} & proximity_{mn} = 2 \\ 0 & proximity_{mn} = 3 \end{cases}$$

where rel_{mfg} is the relationship between group *mf* and group *g*, from the point of view of *mf*, and $effects\_factor_{ng}$ is the importance of group *ng* to the rest of the groups in the playbox. The relationship is a number from from -1.0 to +1.0, where +1.0 indicates that the groups are perfect friends and -1.0 indicates that the groups are perfect enemies. The relationship between a group and itself is always +1.0; the relationships between other groups typically range from -0.6 to +0.6. Note that if the relationship is 0.0, then group *ng* will have no influence at all on group *mf*.

The effects factor is nominally 1.0, but can be adjusted up or down to indicate that a particular neighborhood group has more or less influence. It is always greater than 0.0 and usually less than 2.0.

Given a direct effect on group *ng* with magnitude M_i, then the maximum indirect effect (in absolute terms) on some other group *mf* is $si_{ngmf} \cdot M_i$. Unless $effects\_factor_{ng}$ is significantly greater than 1.0, this usually ensures that the indirect effects are smaller in absolute terms than direct effects.

Indirect effects are further constrained by the near and far factors associated with the satisfaction input; see Section 3.5.6 .

3.5.5 Cooperation Influence

Cooperation influence is similar to satisfaction influence; however, there are significant differences. Cooperation is tracked for neighborhood groups *nf* with force groups *g*. When there is a direct effect on group *nf*'s cooperation with *g*, there is an indirect effect on group *f*'s cooperation with *g* and other force groups, here and in near and far neighborhoods. For example, if *g* does something to antagonize *nf*, thus decreasing *nf*'s cooperation with *g*, then *nf*'s cooperation also decreases with friends of *g*, and increases with enemies of *g*. The same applies for group *f* in other neighborhoods.

For simplicity, we do not model indirect effects on other civilian groups. In other words, if *nf* and *nh* are two neighborhood groups, and are friends, a decrease in *nf*'s cooperation with *g* doesn't affect *nh*'s cooperation with *g*.

The cooperation influence ci_ngmh of a direct effect on some civilian group *f*'s cooperation with force group *g* in neighborhood *n* on *f*'s cooperation with force group *h* in neighborhood *m* is defined as follows:

$$ci_{ngmh} = \begin{cases} rel_{mhg} & \text{if } proximity_{mn} \neq 3 \\ 0 & \text{if } proximity_{mn} = 3 \end{cases}$$

Note that *f* appears nowhere in this definition; since indirect effects are always on the same group *f* as the direct effect, it's only the relationship of force group *h* with force group *g* that matters.

Two comments:

- We do not model group *f*'s perception of the relationship between force group *h* and force group *g*; we simply assume that group *f* knows what it is.
- The GRAM model (and the software that implements it) allows group *h*'s relationship with group *g* to vary by neighborhood. Typically, though, we assume that all force groups are playbox-wide and that relationships between force groups are constant across neighborhoods. The client simulation may enforce this as a constraint.

3.5.6 Here, Near, and Far Factors

The influence factors determine the maximum indirect effect a direct effect may have on some neighborhood and group. However, different drivers differ in scope. For example, accumulation of garbage or raw sewage in the streets of a neighborhood is likely of concern only to the residents of that neighborhood, whereas combat in a neighborhood is almost certainly of concern to residents of near neighborhoods, and possibly of concern to those in far neighborhoods as well. Typically we assume that indirect effects are strongest HERE, decrease in NEAR neighborhoods, and decrease even more in FAR neighborhoods. This is controlled by three parameters associated with each satisfaction and cooperation input: the *here* factor, denoted *s*, the *near factor*, denoted *p*, and the *far factor*, denoted *q*. We require that

$$0.0 \le q \le p \le s \le 1.0$$

The values of *s*, *p*, and *q* can vary from input to input, even for a single driver; however, we usually define *s*, *p* and *q* to be the same for all inputs for each kind of driver. Morever, *s* is usually 1.0.

3.5.7 Scheduling Direct and Indirect Effects

Given the terms defined in the preceding sections, we can now define the direct and indirect effects resulting from an input to GRAM. First, the direct effect is simply a level or slope effect as defined in Section 3.1 . It will a magnitude, which is called the *limit* for level effects and the *slope* for slope effects; here, we'll call it simply *M*.

The indirect effects *j* of a direct satisfaction effect *i* on civilian or organization group *f* in neighborhood *n* will have magnitude

$$M_j = M_i \cdot si_{ngmf} \cdot (s \cdot here_{mn} + p \cdot near_{mn} + q \cdot far_{mn})$$

In other words, the direct magnitude is adjusted by the influence factor, and by *s*, *p*, and *q* as

appropriate.

Similarly, the indirect effects *j* of a direct cooperation effect *i* on group *nf* with group *g* will have magnitude

$$M_j = M_i \cdot ci_{ngmh} \cdot (s \cdot here_{mn} + p \cdot near_{mn} + q \cdot far_{mn})$$

All indirect effects are delayed by $effects\_delay_{mn}$; thus, if the direct effect has start time ts_i and end time te_i, the indirect effects will have start and end time

$$ts_j = ts_i + effects\_delay_{mn}$$
$$te_j = te_i + effects\_delay_{mn}$$

Note that when scheduling slope effects, the bookkeeping associated with slope chains must also be done; see Section 3.1.2.2 .

3.6 History

The ultimate effect of any attitude driver (i.e., an event or situation) is remarkably hard to predict. Because of the effects of scaling and causal analysis (Section 3.1.5) the actual contributions of the driver depend on the current attitude levels, along with everything else going on at the same time. In order to assess the actual contributions of any given driver after the fact, then, GRAM preserves a history of the actual contributions of each driver on each satisfaction and cooperation curve at each time tick. This history table can be used to produce a list of the most significant drivers over a certain period of time, with respect to any particular group or neighborhood.

4. RELATIONSHIP MULTIPLIER FUNCTIONS

A client simulation's attitude rule sets will usually model many kinds of effects that depend on the relationships between acting groups and effected groups. The strength of the effect depends on the relationship *R* between two groups, as mediated by one of a number of *Relationship Multiplier Functions* (RMFs). Given a relationship value *R*, where $-1 \le R \le 1$, each RMF returns a multiplier *r*. This *r* is typically multiplied by the nominal magnitude of an attitude input to produce the actual input.

4.1 Nominal Relationships

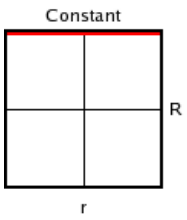
At one time, RMFs were computed such that -1 <= r <= 1; the **Linear**, returned 1 for *R* = 1 and -1 for *R* = -1. Thus, applying an RMF to an attitude input reduced the size of the input for anything but an extreme relationships. Moreover, extreme relationships simply aren't used:[12] the practical range is more like $-0.6 \le R < 0.6$, Thus, applying any RMF but **Constant** to an attitude input was tantamount to reducing the change by a factor of about 0.6. As a result, the change magnitudes shown in the client simulation's rules were misleading. One would expect a nominal change of 10.0, for example, but the actual change would always be smaller. This made it more difficult for casual users to analyze the probable effect of rule firings, and also complicated the process of getting rule inputs from our subject matter experts.

Consequently, each of the RMFs depends on a parameter[13] called $R_{NOMINAL}$, where $0 < R_{NOMINAL} \le 1$. This is the *nominal relationship*, the relationship that the subject matter experts should keep in mind when writing attitude rules. When $R = R_{NOMINAL}$, we expect the RMF to return 1.0 (or -1.0) and thus have no effect on the outcome. As a side effect, RMFs can return values greater than 1.0 and less than –1.0. An RMF can therefore weaken an input, strengthen an input, change its sign, or leave it unchanged, based on the relationship value *R*.

4.2 Specific Relationship Multiplier Functions

Mars defines the following RMFs:

Constant: The "Constant" function returns a constant 1.0, regardless of *R*. It is provided for use in generic code that takes the name of the RMF as an input, to remove the effect of group relationships.

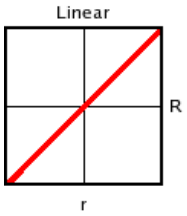


$$r = 1$$

12 Two groups with a relationship of 1.0 might as well be the same group; we usually only see values of 1.0 for the relationship of a group with itself. And a relationship of -1.0 is literally insane. If two groups A and B have a relationship of -1.0 then A is precisely as angered about something good that happens to B as B is pleased about it.

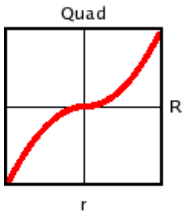
13 Model parameter: `rmf.nominalRelationship`

Linear: The "Linear" function returns a value directly proportional to the relationship. Use this function when the sign and strength of an effect should match the sign and strength of the relationship. The resulting r will have a positive effect on friends and a negative effect on enemies, in proportion to the strength of the relationship.



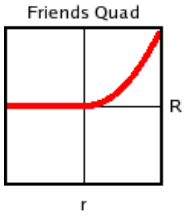
$$r = \frac{R}{R_{NOMINAL}}$$

Quad: The "Quad" function is similar in effect to the "Linear" function; however the resulting effect will be weaker than "Linear" for $|R| < R_{NOMINAL}$ and stronger than "Linear" for $|R| > R_{NOMINAL}$. It is computed as follows:



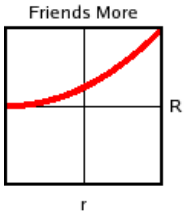
$$r = \left(\frac{R}{R_{NOMINAL}}\right)^2 \cdot \text{sign}(R)$$

Friends Quad: The "Friends Quad" function has an effect that is strong for strong friendships, very weak for weak friendships, and zero for enemies of any degree. Its shape for friendly relationships is identical to the "Quad" function.



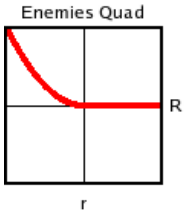
$$r = \begin{cases} \left(\frac{R}{R_{NOMINAL}}\right)^2 & R > 0 \\ 0 & R \le 0 \end{cases}$$

Friends More: The "Friends More" function has a positive effect on both friends and enemies, but friends are affected much more strongly than enemies.



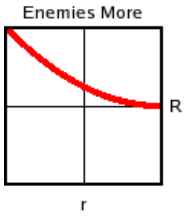
$$r = \left(\frac{1+R}{1+R_{NOMINAL}}\right)^2$$

Enemies Quad: The "Enemies Quad" function has an effect that is strong for strong enemy relationships, weak for weak enemy relationships, and zero for friends of any degree. Its shape for enemy relationships is similar to "Quad", but it does not reverse the sign.



$$r = \begin{cases} \left(\dfrac{R}{R_{NOMINAL}} \right)^2 & R < 0 \\ 0 & R \geq 0 \end{cases}$$

Enemies More: The "Enemies More" function affects both friends and enemies in the same direction, but friends are affected much less than enemies. Like "Enemies Quad", it does not reverse the sign.



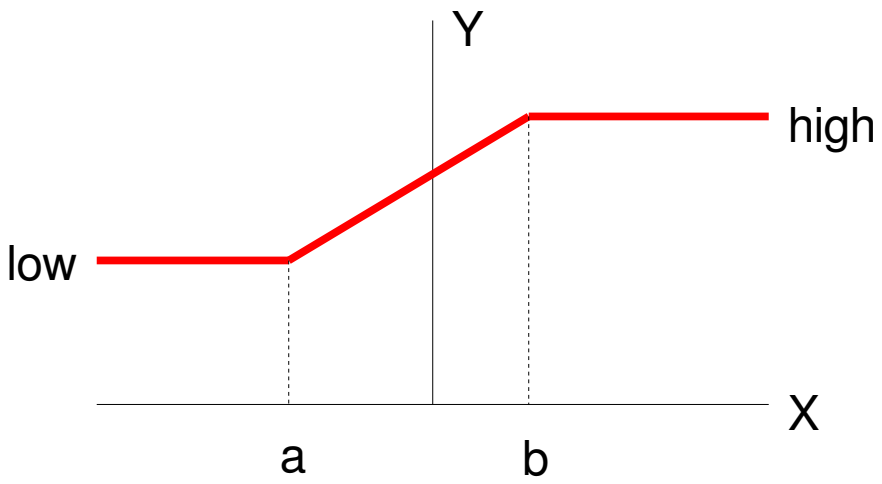
$$r = \left(\frac{1 - R}{1 + R_{NOMINAL}} \right)^2$$

5. MISCELLANEOUS MODELS AND ALGORITHMS

This section documents models and algorithms of general use.

5.1 Z-Curve Functions

A Z-curve function is a stylized S-curve represented as a piece-wise linear curve of three segments. It is defined by the four parameters shown here:



In other words

$$y = Z(x) = \begin{cases} low & \text{if } x \le a \\ low + \dfrac{x-a}{b-a}(high - low) & \text{if } a < x < b \\ high & \text{if } b \le x \end{cases}$$

5.2 Poisson Processes

In a Poisson process, the probability that a single event will occur during *any* very small interval of time is constant, whether other events have occurred recently or not. The average rate of occurrence, λ, determines that probability. The probability that n events will occur during an interval of length t is given by:

$$P_n(t) = \frac{e^{\lambda t}(\lambda t)^n}{n!} \quad \text{where } n = 0,1,2,3,\ldots$$

If the interval of time is always the same, the formula $\mu = \lambda t$ can be used to restate this formula for the probabilities recursively as follows:

$$P_0 = e^{-\mu}$$

$$P_n = P_{n-1} \cdot \frac{\mu}{n} \text{ where } n = 1,2,3...$$

5.3 Selecting a Random Location in a Neighborhood

Use the following algorithm to select a random location in a neighborhood's polygon

Let *tries* = 10.
While *tries* > 0,
    Select a point *p* randomly from the neighborhood's polygon's bounding box.
    If point *p* lies within the polygon, taking overlapping neighborhoods into account,
        Return point *p*.
        Done.
    Otherwise, decrement *tries*.
If no point has been found in 10 tries,
    Return the neighborhood's reference point.