Model Predictive Control of Military Operations

Jan Jelinek and Datta Godbole

Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418

Contact: jan.jelinek@honeywell.com, phone (612) 951-7701

Abstract

Competitors in the marketplace or combatants on the battlefield face very similar challenges: Their resources, be they money or weapons, are gradually attrited in the mutual effort to push each other out of the field and dominate it. Even if the participants are deterministic in their decision-making, executing their decisions has random aspects, when the same, generally successful actions occasionally fail for no obvious reasons. The application of system and control theories to improve the planning as well as the execution of such processes requires models, which allow planners and managers to reliably predict the expected outcomes of various alternatives over a long horizon into the future. In this article, exact probabilistic models for several classes of battle scenarios are developed from the first principles, which accurately characterize the battle dynamics for arbitrarily long horizons. It is shown how the models are used for model predictive control of the battle dynamics.

1. Introduction

Combat is an inherently random process. Viewed as concurrent execution of many combats between individual opponents, battle retains this random aspect, which limits what one can realistically expect from battle models. Every battle is a particular realization of the random processes involved, and if the combatants had a chance to fight it over and over under the same rules of engagement, the outcome would vary from one run to another.

No planning or control is possible unless we can predict the expected outcomes of our decisions into the future. Obviously, the longer is the task horizon, the farther out our predictor must reliably go. Given the randomness of combat, developing reliable predictive models becomes the major design issue.

For almost a century, military theorists have been using the Lanchester model [3] to explain the attrition rates observed in actual battles. Many authors have attempted to fit it to real battle data with varying success. The critique of such endeavors is well summed up in [4,5]. The model is deterministic of the predator-prey kind extensively studied in mathematical biology [10]. It is argued in [4] that due to its determinism it is inherently flawed to capture the reality of combat, and that if there is to be a better model, then the stochastic nature of combat will have to be built into its conceptual fundaments. Another difficulty with using the Lanchester model is the fact that its parameters do not have domain specific meaning and hence need to be tuned by fitting it to battle data. How to set it up before the data become available is an unresolved problem.

Another modeling approach makes use of Monte Carlo models, every run of which would generate one realization of the battle, very much like rolling a die generates one out of the six numbers on its faces. While generally easy to build, the models cannot be directly used for developing battle strategies, nor for amending them on-line when the real time battlefield damage assessment data start coming in. Because strategies are always developed before any actual action takes place, predictive models needed for their design cannot work directly with particular realizations of random variables like the Monte Carlo models do, but only with their distributions or statistics like expectations, variances and so on, which themselves are not random. While theoretically possible, estimating the distributions and statistics from multiple Monte Carlo runs quickly become impractical for larger problems.

The predictive models presented in the paper describe the dynamics of participants' asset degradation over time as a result of combat activities. Due to space limits, we will restrict our explanation to 1-on-1 battles, in which either side deploys only one asset. Modeling complex M-on-N battles is described elsewhere [9]. The models are derived from the first principles and are Markovian in form. Unlike the Lanchester and other models found in literature, our models are exact in the sense that their predictions of battle state probability distributions exactly agree with experimental data furnished by Monte Carlo simulations not just for a couple of combat missions ahead, but for arbitrarily long prediction horizons. This feature makes

them ideally suited for use in the model predictive control framework.

An example of the class of control problems we have been studying may look as follows:

At 0600 the Blue side commander is given the order to completely destroy Red's SAM assets made up of L_R real sites and D_R decoys by 2400 tomorrow. Because the objective is needed to clear the way for an already planned subsequent offensive, the higher command requests the order be executed with a very high degree of certainty, say less than 1 in 20 chances that it will not be met in full. The Red's SAMs are known to have the lethality λ_R against the attacking aircraft that B is intending to use. They also have a good radar tracking capability to know the accurate numbers and positions of attackers in real time.

Our solution answers the following questions:

- How does the Blue commander determine how many attack, L_B , and decoy, D_B , aircraft he needs in his strike package, if his kill rate on the R's SAM's is known to be λ_B ?
- How many missions (sorties), *K*, he should divide his objective into, one, two, or perhaps ten?
- If he decides to fly more missions, how he would define their individual objectives, against which he could measure the task's progress once it gets underway? Without them, he would not be able to identify looming problems until it may be too late for any correction.
- If he decides to fly more missions, how to optimally assemble the strike packages for each one? On one side, gradual enemy attrition will lower the threat, but he will have his losses as well. How big? What is the total number of aircraft he should ask to be allocated for the task?
- If, for whatever reason, the task execution does not proceed as planned, what corrective action to take?

Technically, our solution is a model predictive controller using a probabilistic model and either control- or game-theoretic optimizer.

2. Modeling Battle Dynamics

At any time during a battle, each asset is in a particular state of degradation. The set of all possible states of each asset is considered finite. For example, a fighter squadron's state at a given time is the number of aircraft surviving at that time. Due to random effects present in combat, we are unable to predict with certainty the particular states through which the assets will be passing in the course of battle, but we can derive probability distributions of the assets' states and their evolution in time as the battle progresses.

The state evolution in time is shown to be a Markov chain and thus linear with respect to the state distributions:

$$S(k+1) = T \cdot S(k) \qquad (1)$$

S(k) denotes the $(nLB + nDB + 1) * (nLR + nDR + 1)$ state matrix in the *k*-th round (mission). Formally, the transition matrix has the dimensions $((L_B+D_B+1)$ x $(L_R+D_R+1))^2$, and thus its size quickly grows with the combatants' asset size. However, the matrix is very sparse and if the sparsity is properly taken advantage of, the implementation of (1) leads to very efficient algorithms, both time- and memory-wise, even for battles involving sizeable assets.

From the control perspective, the model's practical value is in the way the user can set up and manipulate its transition matrix *T*. The matrix depends on problem specifications

$$T(L_B, D_B, \lambda_B, L_R, D_R, \lambda_R) \qquad (2)$$

and the theory in [1],which is outlined below, derives an exact functional form of this complicated nonlinear relationship. The fact that our model allows to enter its specifications in a form directly related to militarily meaningful quantities instead of some meaningless mathematical coefficients appearing somewhere in differential equations as is the case with the Lanchester model not only adds transparency to it, but allows us to treat the explicit model parameters as control inputs and vary them on-line during the optimization process inside the MPC controller.

The model treats both sides equally. It accepts as its inputs the parameters (L_B , D_B , λ_B), (L_R , D_R , λ_R). Note that in our model the weapons (aircraft, SAM etc) of B are the targets of R and vice versa. When the model is exercised, it generates a sequence of probability distributions of possible battle states after repeated missions (strikes, rounds). In the sequence illustrated in Figure 1, the plot densities are proportional to the probabilities. The top left picture is the initial state of a battle, when both sides know with probability 1 their initial numbers ($L_B = 4, L_R = 11$). The outcome of the first mission shown in the next picture is not that unequivocal anymore. The most likely number of survivors will be ($L_B = 2$ *or* $3, L_R = 7$), but other outcomes still rather close to those numbers are possible.

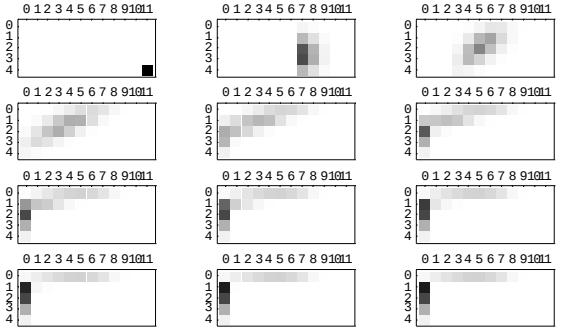

Figure 1: State probability distribution of a battle after 11 rounds (strikes, missions). The win states for B and R are defined as the first column and row, respectively.

As the battle progresses (read Figure 1 row-wise), the cluster spreads more and more until it eventually splits into two, i.e., the distribution becomes bimodal. The last picture

in the bottom right corner says that B will win most battles with 1 or 2 likely survivors, but still there will be a significant share of battles won by R. Whenever that happens, R is most likely to end up with 4 or 5 survivors.

There is one more parameter in the model not yet mentioned, namely the amount and type of real time damage assessment information that the commanders receive. As demonstrated in [1], this parameter has a very strong impact on the battle outcome. Our current models allow the user to choose from the following alternatives:

- Commanders cannot distinguish live targets from dead ones (either killed or decoys).
- Commanders can distinguish live targets from dead ones immediately after each mission.
- Commanders can distinguish live targets from dead ones, but only with time delay amounting to given numbers of missions, which may be different for R and B [2].

Any combination of the above scenarios is also possible, e.g., B gets the feedback information, whereas R does not.

3. Model Predictive Control (MPC) of Battle Dynamics

Known distance to target allows us to estimate the time needed for a single mission and thus put the upper limit, K, on the number of missions that can be flown. We can view K as the control horizon of a batch process controller, which will get shorter with each new round of the battle. Our model predictive controller with the shrinking horizon will try to counter the setbacks caused by bad luck and exploit the windfalls brought about by good luck to drive the battle to victory for Blue within the given deadline. Its optimization criterion formulation is probabilistic: At the beginning of each round, find a platform deployment sequence all the way to the end point which will lead to victory with the probability no less than the desired probability of win. This Victory-at-Any-Cost task formulation [8] ignores the issue of own losses. On the other hand, the Victory-With-Acceptable-Loss formulation [7] strives to satisfy the above criterion in a way that keeps own losses below a given limit.

When searching for optimum, we do not have explore the entire space of all possible deployment sequences of a given length. As it turns out, the so called initial deployment sequences, which require the specified number of airplanes to be all deployed in the current mission and fight the remaining missions only with the survivors of the preceding ones, are either optimal or nearly optimal, even in the game-theoretic sense. This fact has two nice consequences. First, the controller will be a worthy opponent to any intelligent adversary. Second, this feat can be accomplished at a very affordable computation cost.

3.1. Victory-at-Any-Cost Formulation

What is the optimal number of aircraft L_B that B needs to deploy in the first mission, if he knows:

1. the enemy specifications (L_R , D_R ,λ_R),
2. his specifications (L_B , D_B ,λ_B),
3. the rules of engagement direct him to always deploy all surviving aircraft in subsequent missions with no reserves to be added, and
4. the task is to be accomplished in K rounds or less with the desired probability of success $P_{desired}$ or higher?

As the reader may have noticed in the optimization problems above, there was no mention of own losses. Achieving the objective was the only aspect that mattered and if we lost x aircraft along the way, so be it. There certainly are such urgent tasks in wars, but more typically, there will be concerns about losses. Here we are intentionally avoiding the notion of cost, which all classical, deterministically formulated optimization problems introduce as a necessary technicality, but which has a little meaning in war. One may say with some exaggeration that dollars do not make sense in war, and resources and targets do not have their military value price tags painted on them. The probabilistic formulation again offers a more realistic look at the problem.

Once we bring own losses into consideration, the optimization problems take on a different twist. One can easily imagine two extreme alternative solutions to the SAM destruction task cited earlier, and then many others in between. One simply assumes that B takes the enemy's weapon lethality λ_R against his attacking aircraft as a fact of life which he can do nothing about, hence he is going to bite the bullet and use the optimal number calculated by one of the above algorithms. The other is based on assumption that B can actually proactively manipulate R's weapon lethality. For example, adding radar jamming support aircraft to his strike package will reduce SAM's lethality and his losses will drop. If adding one such airplane cuts λ_R by 0.1, how many of them does B need to limit his attack aircraft losses, say, to under 5 pieces and still accomplish the original objective?

3.2. Victory-With-Acceptable-Loss Formulation

What are:

1. the optimal numbers of attack aircraft L_B that B needs to deploy and
2. the minimum reduction of the R's weapon lethality $\lambda_R(k)$ that B needs to achieve in each (k-th) mission from the beginning to end,

if he knows:

1. the enemy specifications (L_R , D_R ,λ_R) for each mission,
2. his specifications (L_B , D_B ,λ_B),
3. the task is to be accomplished in K rounds or less,
4. own losses cannot exceed $loss_B$, and
5. all that must be met with the desired probability $P_{desired}$ or higher?

Here we assume that based on experience, B can translate the required drop in R's lethality into a corresponding number of support aircraft.

In either task formulation, the criterion is minimized under the initial deployment sequence assumption by a simple branch-and-bound algorithm. The foe R is modeled as either strictly following a fixed set of rules of engagement known to B (control-theoretic formulation), or can be an intelligent adversary always trying to deploy his resources so that he harms B the most (game-theoretic formulation).

3.3. Closed Loop MPC

No feedback control, also called open loop control by control theorists, computes the optimal deployment of forces only once, before the task execution starts. Once it gets going, it lets its sequence of missions run its own course, without any further intervention. Commanders simply keep sending back into action all surviving aircraft until one side looses all. It was this kind of control, or rather lack of it, that was used to produce the plot in Figure 1.

Now imagine that B has some reserves that he can bring in if bad luck drives him off the planned course. On the other hand, when luck has him do better than anticipated, then he can put the unneeded aircraft back into the reserve pool to make it available to others rather than finishing his task ahead of the plan. Using one of our predictive model based optimizations listed above, we can now easily build a model predictive controller (MPC) of battles. Despite its simplicity, the MPC performs extremely well and shows considerable robustness with respect to model-plant mismatches.

We shall explain the Initial Deployment MPC concept by contrasting it against the open loop (= no control) solution.

Prior to the first mission, both the no-control and MPC-control strategies do the same calculations, namely compute the optimal number of aircraft to be flown in the first round (mission) using the initial deployment optimizer. Even though the optimizer explicitly returns only one number, namely the number of aircraft to be used in the first mission, it actually computes the solution all the way up to the victorious endpoint assuming that all survivors will always be redeployed in full. If B wants to know the expected losses in each round, both his own and R's, he can obtain them easily by exercising the predictive model using the optimal number L_B.

Because both strategies use the same optimization algorithm, they come to exactly the same conclusions. Therefore, the first mission strike package makeups are always identical. After the first mission, however, they will start to differ. The no-control commander will thoughtlessly gather all his surviving resources and order them to fly the second mission. The MPC-controller (or, better, the MPC-advised commander) will first look at the damage assessment intelligence and critically reevaluate his standing. For this purpose he will use the same model as before, but will enter the intelligence updates on the actual enemy strength after the first mission and will also reduce by one the maximum number of missions allowed to fulfill the task objective. This reevaluation will generally produce slight corrections to the actual number of survivors that the no-control commander will use, because our first mission plan could not know what would exactly happen in combat and thus worked only with outcome probabilities. Feedback is thus closed through the ongoing replanning and implemented as corrections to package composition. The corrections are either drawn from or returned to the pool of reserves.

3.4. Experimental Results

In this section we present results of numerical experiments where the Blue MPC controller was driving a Monte Carlo battle simulator fighting the Reds, whose commander was dutifully following his orders to always redeploy all survivors. Vertical bars represent the reserves added to or withdrawn from the survivors. The first mission bars are the initial deployments, and thus must be identical in all battles. There are no other red bars, because R makes no follow-up corrections to the survivors. The thin lines are the number of survivors after each mission, the dots mark the actual number of aircraft deployed. Note that the red dots always lie on the same horizontal level as was crossed by the thin survival plot in the preceding mission. Due to his ability to add or withdraw airplanes, this is not generally true for B, for whom the sum of previous mission survivors and reserve change in the current mission determines the blue dot's vertical position. Figure 2 plots average strength from the statistics collected on one batch of 1000 randomly generated battles fought under the same scenario as before. The most remarkable observations are that:

1. MPC did not loose a single battle, and
2. on average, it kept withdrawing rather than adding reserves.

The latter indicates that in addition to the outstanding performance, one of the rather unexpected benefits of MPC control is a better utilization of resources.

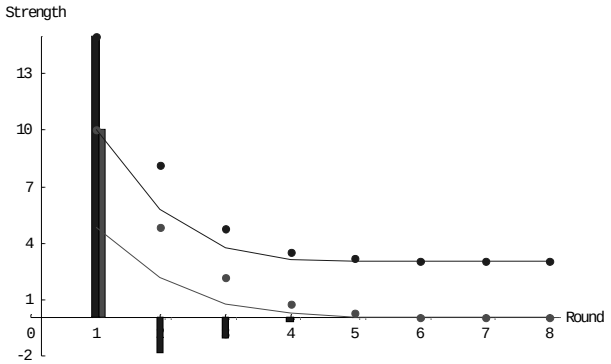

Figure 2: Averages from an ensemble of 1000 randomly generated experimental battles. Bars, dots and solid lines represent deployment increments/decrements, actually deployed and surviving aircraft in each mission, respectively. Lacking color, the plots loose some of their explanatory effect.

As the following Figure 3 illustrates, both the MPC-controlled and no-control battles have losses identical within the statistical margin error. For comparable losses, however, we get very different return on our resource

investments. The average numbers of deployed aircraft per mission without control and with MPC control plotted in the Figure 4 convincingly demonstrate the wasteful use of resources in the no-control case. This contrast is further amplified by the fact that while the MPC battle controller did not loose a single battle in the ensemble of 1000 (although it may very rarely happen), with no control the probability of losing is not even close to zero. That is, without control, on average, we can expect either to loose or not win within 10 rounds about 66 battles in every ensemble of 1000.

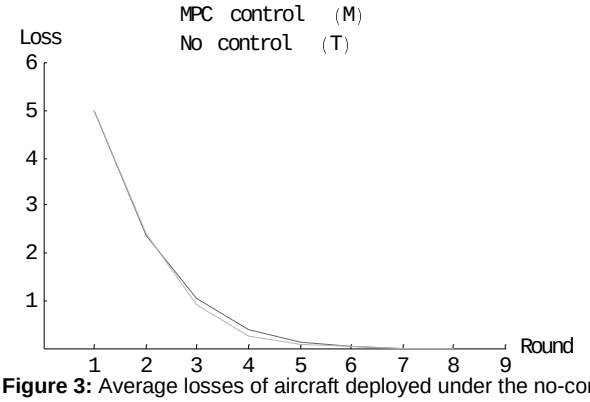

Figure 3: Average losses of aircraft deployed under the no-control (lower plot) and MPC (upper plot) strategies.

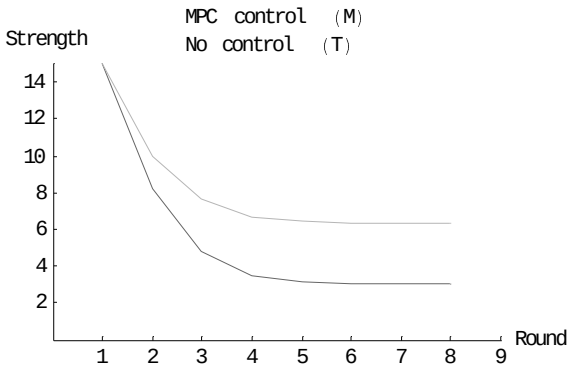

Figure 4: Average numbers of aircraft deployed under the no-control (upper plot) and MPC (lower plot) strategies.

4. Robustness of the MPC Battle Controller

Robustness refers to the resilience of the model predictive controller performance to mismatches of its internal predictive model with the reality. We have experimentally investigated two kinds of plant-model mismatches:

1. The MPC model under- or overestimates the lethality of R assets,
2. The B's real time damage assessment intelligence is inaccurate and under- or overestimates the size, L_R, of R assets.

Results from thousands of Monte Carlo simulations of various degree of mismatch convincingly demonstrate remarkable immunity of the MPC battle controller to inaccurate battlefield information. However, as with everything else in life, the ignorance does have its price: The MPC controller will keep winning, but B will pay for his victories with either increased losses or higher opportunity costs for wastefully using his aircraft to no extra benefit. The following subsections offer the gist of the results.

4.1. B Underestimates the Lethality of R Assets

Figure 2 through 4 illustrate the performance of the MPC controller, which plays the role of the B commander, providing that its model of R perfectly matches his actual strength and lethality. Below in the Figure 5 results are presented for the case when the actual lethality is $\lambda_R = 0.5$ as before, but B believes that it is only $\lambda_{Rm} = 0.2$.

In this particular ensemble of 1000 battles, B happened to lose 4 of them, which seems to be a still acceptable performance degradation. This number can slightly fluctuate for other ensembles due to the random nature of Monte Carlo simulations, but generally will be in the range of a couple of losses. As we can see in Figure 5, B starts with only 10 airplanes in the first mission, but his incorrect lethality estimate forces him to bring in more and more reserves in each of the subsequent missions and, at the end, pay for his error with higher expected losses. It is interesting that they are higher in spite of the total number of deployed aircraft being lower. Also note that the number of missions B needs to win goes up.

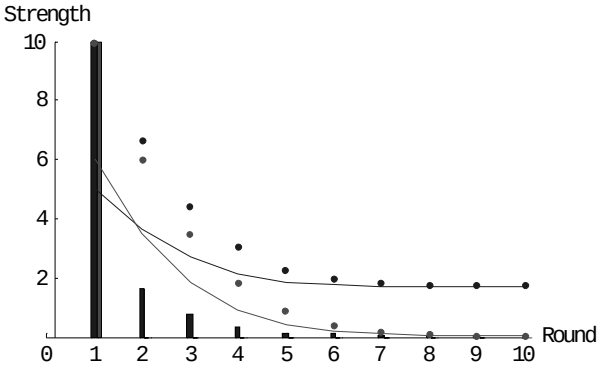

Figure 5: Averages from an ensemble of 1000 randomly generated experimental battles, in which B mistakenly guesses the R weapons' lethality at 0.2, while it actually is 0.5. B and R strengths are the upper and lower plots.

We can better appreciate the excellent MPC performance once we realize that the loss of only a couple of battles out of 1000 is nothing compared to the disaster which such a gross underestimation would have led to without control. In that case, the probability of B loosing is 0.747802, i.e., B can expect either to loose or not win within 10 rounds about 748 battles in every ensemble of 1000.

4.2. B Underestimates the Strength of R Assets

Underestimating the enemy's strength has similar effects as underestimating his weapons' lethality. In each

subsequent mission, B has to keep adding airplanes from his reserves in his attempt to meet the task objective (see Figure 6). In each mission, his plan undergoes significant revisions, and yet he never quite catches up with the reality. As the statistics below show, if his damage assessment underestimates R's numbers by 50%, then he would end up loosing 51 battles out of 1000. (There was one draw in this particular ensemble). As bad as it looks, it is actually a testament of the excellent controller performance. Without MPC control, B would have lost about 952 battles in every ensemble of 1000 on average. He pays a premium for his victories, though, in terms of increased own losses compared to the full information scenario. This shows the importance of intelligence in winning a battle.

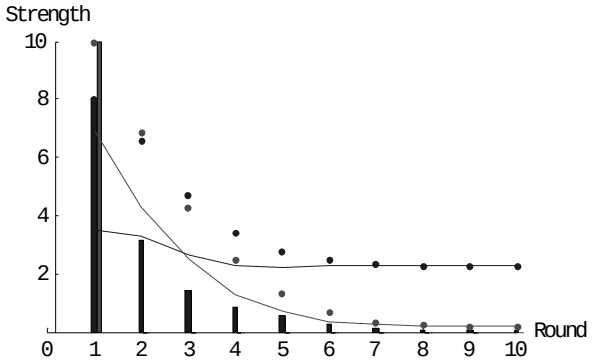

Figure 6: Averages from an ensemble of 1000 randomly generated experimental battles, in which B receives damage assessment intelligence underestimating the R's surviving numbers by 50%.

Similar results [8] are obtained when B overestimates enemy lethality or numbers, except that in this case MPC starts out with excess deployment of resources and ends up withdrawing many of them as the battle progresses.

5. Summary and Future Work

We presented the design and experimental verification of the MPC Task Commander, which enables the commander to conduct the battle associated with a given task so as to achieve its military objectives with the desired certainty and within the given deadline. As we have demonstrated in thousands of Monte Carlo experiments, the controller hardly looses a battle, even if its information about the enemy is not particularly accurate. Because its victories are always accomplished with the minimum number of resources needed for the given task, its application in concurrent tasks increases the utilization level of military resources by allowing them to be deployed in tasks where they have the biggest impact. The MPC performance has been shown to be remarkably robust with respect to the reality-model mismatches. Our experiments also show the importance of obtaining correct battle damage assessment and intelligence information.

In addition to control, the models can be used in other related applications as well. We have researched the following topics with positive results:

- Model-based Damage Assessment Intelligence Evaluation (to be published)
- Resource Apportionment among Concurrent Tasks [6]
- Controlling Battles Against Intelligent Adversary (to be published)

Acknowledgements: This document is based upon work supported by the DARPA JFACC program through SPAWAR Systems under contract no. N66001-99-C-8501. Any opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of DARPA and SPAWAR Systems.

Bibliography

1. Jelinek, J. " Predictive Models of Battle Dynamics", JFACC project report, Honeywell HTC, February 2000
2. Jelinek, J. " Predictive Models of Battle Dynamics: Part II - Delayed Feedback", JFACC project report, Honeywell HTC, April 2000
3. Hillestad, R. J. and Juncosa, M. L.: " Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models", in Bracken, J. et al, eds.: Warfare Modeling, John Wiley & Sons 1995
4. Ancker, C. J., Jr.: " A Proposed Foundation for a Theory of Combat", in Bracken, J. et al, eds.: Warfare Modeling, John Wiley & Sons 1995
5. Hartley III, D. S.: "A Mathematical Model of Attrition Data", in Bracken, J. et al, eds.: Warfare Modeling, John Wiley & Sons 1995
6. Tierno, J. "Distributed Multi-Battle Model Predictive Control", JFACC Project Report, Honeywell HTC, June 2000
7. Jelinek, J. "Winning Battles Without Losses", JFACC Project Report, Honeywell HTC, June 2000.
8. Jelinek, J. and Godbole, D.: "Model Predictive Control of Battle Dynamics", JFACC Project Report, Honeywell HTC, May 2000.
9. Jelinek, J.: "Modeling Complex Battles", JFACC Project Report, Honeywell HTC, September 2000.
10. Epstein, J.M.: "Nonlinear Dynamics, Mathematical Biology, and Social Science", Lecture Notes Vol. IV, Santa Fe Institute, Studies in the Sciences of Complexity, Addison-Wesley 1997