# Real-Time Fatigue Monitoring with Computational Cognitive Models

Leslie M. Blaha[1(✉)], Christopher R. Fisher[1(✉)], Matthew M. Walsh[2], Bella Z. Veksler[1], and Glenn Gunzelmann[1]

[1] Air Force Research Laboratory, Wright-Patterson AFB, OH, USA
{leslie.blaha,christopher.fisher.27.ctr,glenn.gunzelmann}@us.af.mil, bellav717@gmail.com
[2] Tier1 Performance Solutions, Covington, KY, USA
mmw188@gmail.com

**Abstract.** Real-time monitoring with cognitive models offers the unique ability to both predict performance decrements from behavioral data and identify the responsible cognitive mechanisms for targeted interventions. However, their potential has not been realized because current parameter updating methods are prohibitively slow. We present a paradigm that enables real-time monitoring using cognitive models and demonstrate its implementation with a fatigue-sensitive task. In this demonstration, an operator workstation, a cognitive model, and a monitoring station are networked such that task performance data are sent to a central server that estimates model parameters and generates model-based performance metrics. These are sent to a monitoring station where they are summarized graphically together with model fit diagnostics. This constitutes an infrastructure that can be leveraged for future predictive adaptive system designs.

**Keywords:** Cognitive augmentation · Real-time monitoring · Parameter estimation · Fatigue · ACT-R · Computational cognitive models

## 1 Introduction

We envision a future wherein computational cognitive models are employed in real time to monitor and predict operator performance [9]. It is widely acknowledged that cognitive moderators, like fatigue, stress, and task load, can negatively impact performance (e.g., [7]). Naïve approaches to mitigating these moderators involve simple observations, whereby an intervention is administered, usually by a human observer, after a performance decrement is detected. This method suffers from two critical shortcomings. The first is that simple observation is not predictive. An intervention is administered only after performance is already deteriorating and, in some sense, after it is too late. Consequently, poor performance is inevitable. A second shortcoming is that simple observation is purely descriptive, not prescriptive. An intervention—such as resting, altering the task, or recommending a new task strategy—cannot be prescribed because


D.D. Schmorrow and C.M. Fidopiastis (Eds.): AC 2016, Part I, LNAI 9743, pp. 299–310, 2016.
DOI: 10.1007/978-3-319-39955-3_28

the reason for performance decrements is unknown. Without recognizing the source of the performance decrement, there is little basis for selecting appropriate cognitive augmentations. A theoretically grounded method is needed for monitoring the human cognitive system so that performance decrements can be predicted and mitigated through appropriate, targeted interventions.

We propose using cognitive models to assess cognitive states in real time. As mathematical and computational instantiations of cognitive theories, cognitive models offer the ability to make inferences about the processes underlying task performance and how those processes respond to cognitive moderators. Changes in key model parameters representing affected cognitive mechanisms reflect fluctuations in moderating factors. When the cognitive mechanisms are captured in this way, performance decrements can be understood, predicted, and mitigated through theoretically-informed interventions. However, heretofore, the explanatory power of cognitive models has been limited to post-task assessment because computationally intensive simulation is required to fit a model to data. This process can require hours or even days to complete.

In this paper, we present a novel paradigm enabling real-time cognitive modeling for performance monitoring and demonstrate its application to real-time fatigue monitoring with a model specified in the cognitive architecture Adaptive Control of Thought–Rational (ACT-R) [1]. Our demonstration consists of an operator workstation recording someone's behavior, a server implementing real-time parameter estimation, and a (possibly remote) monitoring station summarizing performance metrics. This three-part architecture leveraging networked computational resources is one possible design enabling remote, portable cognitive model-based performance monitoring.

## 2 Parameter Estimation

### 2.1 Standard Methods

A variety of methods are currently available for estimating model parameters [14]. These methods include grid search, gradient descent optimization algorithms (e.g., simplex algorithm, simulated annealing), and various algorithms inspired by biological evolution. One commonality underlying these methods is a trade-off between speed and accuracy. Higher estimation accuracy can be achieved by searching more of the parameter space, but this is costly as it requires more time and computational resources. Even with the benefit of parallel computing, these methods are prohibitively slow and offer limited scalability to complex models. This computational bottleneck has two primary causes: (1) simulations are computationally intensive, and (2) similar areas of the parameter space may be searched repeatedly. The resulting speed-accuracy trade-off is at odds with a fundamental requirement of real-time monitoring: parameter estimation must be fast *and* accurate, not a compromise between the two. This problem is further exacerbated by the need to continually update parameters as new data become available. Thus, the major challenge for real-time monitoring

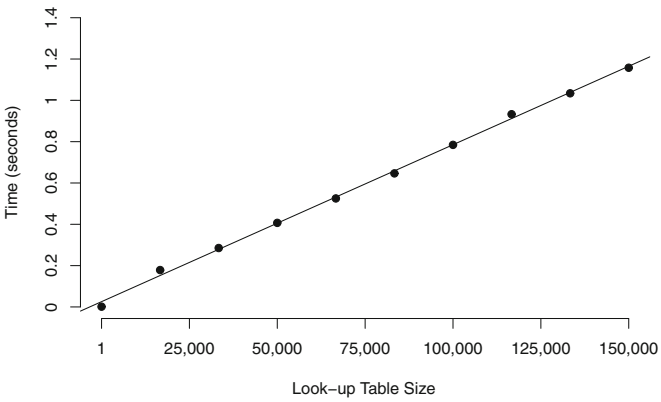


**Fig. 1.** Total parameter estimation time in the PDLT method as a function of look-up table size. Each data point represents the mean of 50 look-up repetitions.

with cognitive models is minimizing the parameter estimation speed-accuracy trade-off.

### 2.2 Pre-computed Distributed Look-Up Table

Our approach eliminates the aforementioned bottleneck through the use of pre-computation and distributed computing, resulting in efficient parameter estimation suitable for simulation-based models. In our method, incoming data from the experiment are compared to model predictions stored in a distributed look-up table with the aid of distributed computing. The parameters corresponding to the best matching predictions are selected as the best-fitting parameters. Accordingly, we termed this method the Pre-computed Distributed Look-up Table (PDLT) method. As shown in Fig. 1, the PDLT method is fast and exhibits a linear increase in computing time as the look-up table size increases. For the relatively simple, 4-parameter model used herein (described below), the PDLT consisted of 150,000 values, and PDLT search required little more than 1 s to complete. By contrast, parameter estimation for the same model using a simplex algorithm implemented on the same hardware can take 1 to 40 min, depending on several factors, including the numbers of starting points, iterations, simulations per evaluation, the speed of the model (complexity), and the model parameter values which scale to the predicted reaction times. The PDLT method has the additional advantage that it easily scales up to slower, more complex models: after the PDLT is constructed, the look-up process is invariant to the time required to simulate the model.

The PDLT method involves three steps, illustrated in Fig. 2:

1. **Define the distribution of parameters, $\Theta$.** The first step is to define a distribution over the allowable parameter space, $\Theta$, from which parameter combinations, $\theta$, are sampled. The distribution's dimensionality reflects the

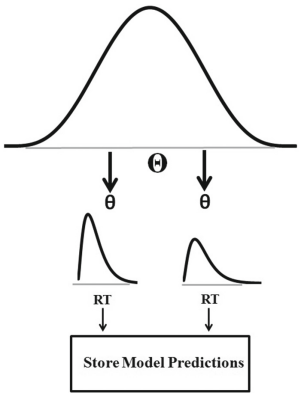


**Fig. 2.** A diagram of the PDLT method. For simplicity of this drawing, $\Theta$ is a unidimensional space, depicted by the Gaussian distribution. Individual samples $\theta \in \Theta$ make predictions for reaction time (RT) distributions.

number of parameters in the model; the shape and range of the distribution encodes which combinations of parameters are likely to be consistent with human behavior. Information from relevant psychological theory and empirical parameter estimates from prior studies are used to inform the construction of this distribution.

2. **Simulate predictions for each parameter combination, $\theta$.** The second step is to simulate model predictions for each parameter combination, $\theta$, and to summarize the predictions with a statistic. As shown in Fig. 2, the model we use for this demonstration generates reaction time (RT) distributions, but other models might generate different behavioral outputs. Other possible statistics include proportions, means, quantiles, or kernel density functions. What is important is that the statistic is a sufficient estimator, which means that the statistic contains all of the information for parameter estimation available in the raw data. The fit of the model is assessed by comparing the statistic computed from the model prediction to the corresponding statistic computed from the empirical data by means of root mean squared error (RMSE) or maximum likelihood.
3. **Store data in the look-up table.** The final step is to store the parameter combinations, $\theta$, and corresponding predictions in a distributed array. Once the distributed array is constructed, it can be saved and re-used for future applications of the same model.

## 3 Real Time Fatigue Monitoring

Now that we have outlined the PDLT method for real-time parameter estimation, we describe its application for real-time fatigue monitoring. Figure 3 provides a generic schematic of the information flow in our demonstration. Data flows from

the left to the right through networked computational resources. Beginning on the left, an operator executes a task at a workstation. Data are continuously transferred from the workstation to a central server, where the PDLT method updates model parameters after every new data point arrives. The server consists of a cluster of computers that perform the PDLT method and generate empirical and model-based performance metrics. Finally, the metrics and model output are transferred to a monitoring station where a human supervisor can observe performance through a summary dashboard. The dashed arrow connecting the server to the operator workstation represents potential for additional cognitive augmentation/intervention. When the model predicts performance decrements, it could, in principle, send feedback to the user or communicate to software designed to mitigate fatigue in affected cognitive systems (e.g., motor or visual modules). In the following sections, we describe a fatigue-sensitive task and corresponding ACT-R model, as well as the fatigue monitoring metrics and components of a supervisory dashboard.

### 3.1 Psychomotor Vigilance Task

We demonstrate real-time fatigue monitoring with the psychomotor vigilance task (PVT) [6]. The PVT is a simple detection task sensitive to fatigue due to sleep deprivation, circadian rhythm variations, and extended time on task [2]. Within each trial, operators wait a random inter-stimulus interval (ISI) of 2–10 s for a visual cue to appear on a computer screen or display device. Once the cue is detected, operators respond by keystroke as quickly as possible. After a response is submitted, RT feedback is displayed for one second, and then a new trial begins. Performance on the PVT exhibits predictable changes as operator fatigue increases. First, the RTs get slower, resulting in an RT distribution with a longer right tail. Second, the number of lapses (missed responses or RTs $>$ 500 ms) and false starts (responses before stimulus onset or RTs $<$ 150 ms) both

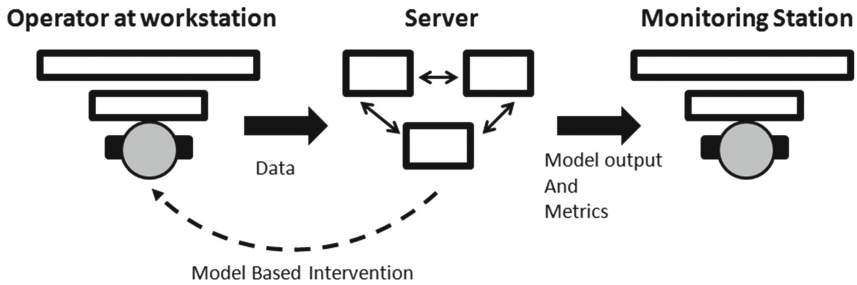


**Fig. 3.** A diagram of the real-time fatigue monitoring demonstration. The operator on the left performs the PVT task at a computer workstation, while a separate individual monitors performance from a remote monitoring station (right-hand side). The two individuals need only be networked to the central server. The dashed line back to the operator represents possible feedback or intervention strategies not implemented in the current demo.

increase. The PVT is typically run in 10 min intervals; shorter versions of the PVT (5 min and sometimes as little as 2 min) have also been demonstrated to show sensitivity to fatigue during sleep deprivation [13]. However, fatigue effects are not confined to instances in which subjects are sleep deprived. After 35 min of performing the PVT, performance decrements can be as pronounced as those found in operators who were awake for 29 h [15].

### 3.2 ACT-R Fatigue Model

Gunzelmann and colleagues [8] developed a model in the ACT-R cognitive architecture to capture performance in the PVT under different degrees of fatigue. ACT-R is a cognitive architecture or computational framework for instantiating comprehensive theories of cognition [1]. The ACT-R cognitive architecture contains a set of information-processing modules, each specializing in a specific mechanism, such as vision, declarative memory, or motor execution. Each of these modules is connected to a centralized procedural module that selects and regulates the internal actions of the architecture. Decision making behaviors emerge over a sequence of production cycles. A production cycle begins when the internal state of the cognitive architecture and environment are compared to the production rule selection criteria. Much like an if–then statement, a production rule is enacted when the conditions of the architecture or environment match its criteria. After a production rule is enacted, a new state is realized, marking the beginning of the next production cycle.

The ACT-R model of PVT performance is based on three productions [8,16]: (1) wait, (2) attend, and (3) respond. The model decomposes the process into four primary parameters: Utility $U_S$, Threshold $U_T$, fatigue decrement $FP_{dec}$, and *cycle time*. On each production cycle, one of the three productions is selected stochastically based on its match with the state of both the environment and the cognitive architecture. The match is formalized with the following utility function:

$$U_{ij} = U_S(U_i - MMP_{ij}) + \epsilon. \tag{1}$$

$U_{ij}$ is the utility of production $i$ in state $j$, $U_S$ is the utility scalar, $U_i$ is the stored utility for production $i$, $MMP_{ij}$ is the mismatch penalty for production $i$ in state $j$, and $\epsilon$ is logistically distributed noise. The mismatch penalty, $MMP_{ij}$, ensures that the incorrect production is selected with a low probability. Behaviorally, this manifests as infrequent false starts and lapses. The parameters $U_S$ and $U_T$ jointly determine the production selection probability. The production with highest utility is selected and enacted if its utility exceeds $U_T$:

$$\text{Production} = max(U_{ij}) \text{ if } max(U_{ij}) > U_T. \tag{2}$$

The difference between $U_S$ and $U_T$, denoted as *Diff*, is an important indicator of fatigue. As *Diff* decreases, the selection of production rules becomes more stochastic. This increases false starts and lapses in the model's behavior. When none of the production utilities exceed $U_T$, a microlapse occurs. On the subsequent production cycle, $U_S$ is decremented according to the parameter

$FP_{dec}$: $U_S = U_S \cdot FP_{dec}$. The likelihood of microlapses increases in subsequent production cycles and can eventually culminate in a behavioral lapse. The parameter *cycle time* governs the duration of production selection at the beginning of each production cycle. The production cycles' summed durations result in the observed RT.

In creating the PDLT, we used maximum likelihood estimation to find the best fitting parameters for a given set of data. This entails adjusting the parameters until the probability of the data given the model is maximized. Because no analytic solution for the likelihood function exists for this ACT-R model, we approximated the underlying distribution with a kernel density function. This method fits a smooth function to a distribution of simulated data. The kernel density estimator interpolates the probability density of a target RT by weighting simulated RTs as a decreasing function of distance from the target value. Goodness-of-fit was assessed with the log likelihood fit statistic, formed by summing the natural log of the predicted probabilities across all responses.

### 3.3 Software and Hardware

Communication between the operator workstation, server, and monitoring station was mediated through websockets. We implemented the PDLT method in Julia, a fast, high-level scientific programming language freely available through an MIT license [3]. The PVT and dashboard were programmed in JavaScript to maximize portability, and the plots were generated with the D3 library [5]. The server implementing the PDLT consisted of four Mac Pro desktop computers consisting of sixteen 3 GHz cores and 32 GB RAM.

### 3.4 ACT-R PVT Model PDLT

1. **Define the distribution of parameters, $\Theta$.** We used multivariate Gaussian distributions based on parameters from individuals who were well-rested and sleep-deprived for 72 h [16]. The multivariate Gaussian distribution ensures efficient sampling of the parameter space by taking into account the central tendency and covariance structure within the empirical distribution of parameters.
2. **Simulate predictions for each parameter combination, $\theta$.** We sampled 150,000 parameter combinations to achieve a high degree of accuracy while also maintaining speeds necessary for trial-by-trial updating of the model. Half of the 150,000 parameters were sampled from a parameter distribution representing well-rested individuals, and the remaining half were sampled from a parameter distribution representing sleep-deprived individuals.
3. **Store data in the look-up table.** Consistent with Fig. 1, the PDLT for 150,000 parameters requires 1.2 s to search, which fits within the minimum 2 s ISI window for the PVT task.

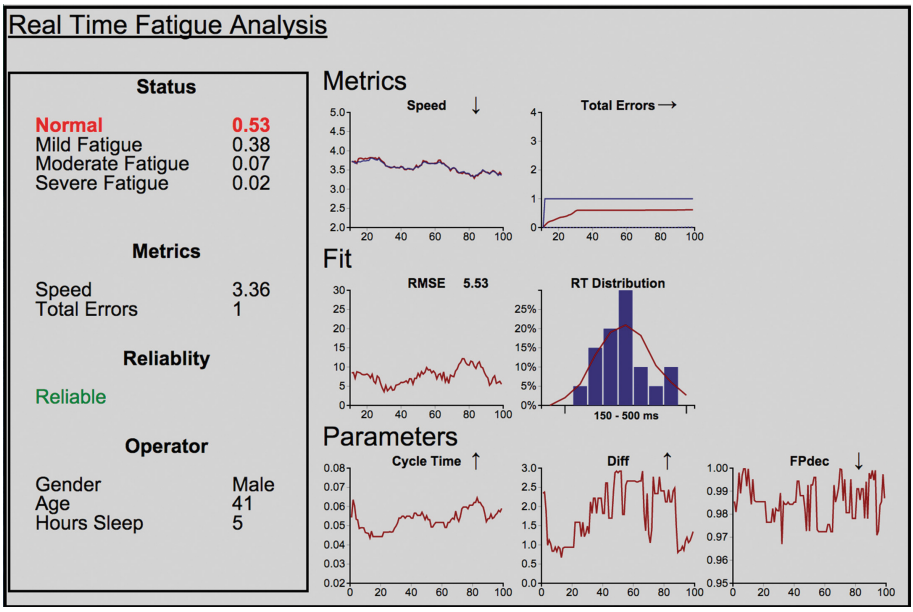


**Fig. 4.** PVT Metrics Dashboard. The data contained herein include 100 PVT trials. Human performance is plotted in blue, and model data are plotted in red. X-axes are the number of trials in all plots, except for the RT Distribution whose axis is RT. Arrows within each plot title indicate whether the last 10 values are on an increasing (↑), decreasing (↓) or stable trend (→). (Color figure online)

### 3.5 Performance Monitoring Dashboard

As a demonstration, we selected 100 trials from one participant who completed an extended PVT session [15] and sequentially fed this data into the fatigue monitoring system. Trial-by-trial, the ACT-R PVT model was fit to the data, and the dashboard received updated metrics and model output. The metrics and model outputs were computed from a 20-trial moving window. A screen shot of the dashboard is shown in Fig. 4, illustrating the data after the entire 100-trial session was completed. Information streamed into the dashboard is broken down into three categories explained in further detail below. In addition to three types of data about the operator and model, a key to the dashboard is presenting principled inferences about the operator state and model reliability for rapid assessment. Herein, we propose an initial set of heuristics for fatigue monitoring, which are summarized in the right-hand panel.

**Operator Performance Metrics.** PVT performance can be tracked by diagnostic metrics sensitive to the effects of fatigue, which are reviewed by Basner and colleagues [2]. While a number of metrics have been proposed, they are all variations on response speed, number of lapses, and number of false starts.

According to Basner et al., the two metrics most sensitive to acute total sleep deprivation are (1) the total errors, defined as the number of lapses and false starts together, and (2) speed, defined as mean(1/RT). The latter is also sensitive to chronic sleep restriction. While we include these in our dashboard, ultimately, the choice of metrics is left to individual applications and operator needs.

The top Metrics row shows the response speed, mean(1/RT), in the left plot. Operator RTs are blue, and model RTs are red. In this example, the two sets of data are nearly identical after 30 trials. The down arrow indicates that speed is on a decreasing trend after 100 trials. The right plot, Total Errors, tracks both the number of lapses (solid lines) and the number of false starts (dashed lines). Note that in this data set, there were no false starts, so that line is on the 0 value. The human data (blue) produces a step function when an error occurs; the model (red) is more graded as it represents the expected number of errors over time. The flat arrow shows a steady trend in the human performance after 100 trials.

**Model Fit.** The validity of the model assessment depends critically upon the fit of the model to the data. A large discrepancy between the data and model prediction will lead to invalid inferences. Accordingly, Fig. 4 displays two fit assessment plots in the middle row. The left plot tracks RMSE as a function of trial. Although maximizing the log likelihood has desirable statistical properties [4], it has poor interpretability due to log-scaling. To improve interpretability, the dashboard displays RMSE between the empirical and predicted RT quantiles. The quantiles correspond to the $10^{\text{th}}, 20^{\text{th}}, \ldots, 90^{\text{th}}$ percentiles, excluding any false start RTs to reduce volatility. One attractive property of the RMSE is its intuitive scaling: RMSE represents the standard deviation between observed and predicted values in milliseconds. A large value for RMSE could indicate either that the model is not valid for the task or that the parameter space was not properly sampled when constructing the PDLT. Should an application of the PDLT technique find unacceptably large RMSE values, *post hoc* model analysis is recommended to determine which of these possibilities resulted in a misfit and to improve the PDLT as needed.

The RT Distribution plot illustrates the histogram of operator RTs (blue) with the best-fitting model prediction overlaid in red. Only valid RTs are shown, in the range of 150–500 ms, excluding false starts and lapses. This provides a fast visual assessment of the similarity in the two distributions in a way that would highlight any strong differences in central tendency or shape.

**Model Parameters.** In the bottom row of Fig. 4, the dashboard tracks the history of values for *cycle time*, *Diff*, and $FP_{dec}$ throughout the task. We opted to track these particular parameters because *Diff* is the most sensitive indicator of fatigue; *cycle time*, although invariant to fatigue, is an indicator of individual differences in baseline speed, and lower values of $FP_{dec}$ correspond to an increased likelihood of micro-lapses.

**Summary Panel.** In time critical situations, integrating and interpreting the data may require more time than is available. For this reason, the dashboard provides a summary panel (left side) displaying the most pertinent information to facilitate rapid fatigue assessment. Starting from the top, the summary panel includes a status indicator, highlighting the operator's most likely level of fatigue in red (detailed below). Next, the summary panel displays the current state of the performance metrics. A reliability indicator located below the metrics provides a binary classification of the quality of model fit. Here, "Reliable" is a green font if RMSE $< 15$ ms, and "Unreliable" is a red font when RMSE $\geq 15$ ms.[1] Finally, the summary panel includes demographic information about the operator, which can be useful when monitoring multiple operators.

The status indicator is designed to distill the available data into one of $N_s = 4$ ordinal fatigue categories: normal, mild fatigue, moderate fatigue, and severe fatigue. We used a Bayesian classifier to categorize performance into ordinal fatigue categories according to the composite parameter *Diff*, which is the strongest indicator of fatigue in the ACT-R model. Our rationale is that model fitting serves as a data reduction method that maps a full distribution of behaviors—including the aforementioned fatigue metrics—onto a small set of meaningful parameters. Because parameters associated with similar levels of fatigue tend to cluster together, they can be further distilled into categories. The intuition behind our application of the Bayesian classifier is that it measures the degree to which an estimated value of *Diff* is consistent with various categories whose underlying dimensions are graded and partially overlapping. Thus, the results of the Bayesian classifier should be interpreted as a category probability rather than a posterior probability of the ACT-R model. Formally, the Bayesian classifier is defined as:

$$\Pr(S_i|\hat{\omega}_t) = \frac{Pr(\hat{\omega}_t|S_i)Pr(S_i)}{\sum_{i=1}^{N_s} Pr(\hat{\omega}_t|S_i)Pr(S_i)}, \tag{3}$$

where $S_i$ is fatigue category $i$, and $\hat{\omega}_t$ is the maximum likelihood estimate of *Diff* on trial $t$. The likelihood of the $\hat{\omega}_t$ given category $S_i$ is defined as:

$$\Pr(\hat{\omega}_t|S_i) = \phi(\hat{\omega}_t; \mu_i, \sigma_i) \tag{4}$$

where $\phi$ is the Gaussian distribution probability density function, $\mu_i$ is the mean, and $\sigma_i$ is the standard deviation. We eliminate the influence of the prior probabilities $Pr(S_i)$ by setting them equal to $\frac{1}{4}$.[2] Each category of fatigue is represented as a Gaussian model corresponding to a distribution of parameters that typify a given level of fatigue. The distributions of *Diff* were formed from the 35 min PVT study using the 17 individuals who exhibited time-on-task fatigue [15]. Fatigue

[1] It is important to note that this threshold is intended as a rough guideline for assessing this model fit for the PVT specifically. For PDLT-based monitoring with other models or tasks, RSME thresholds should be appropriately assessed and tailored.

[2] Updating priors in this context is problematic because parameters are estimated from overlapping windows of data, resulting in non-independence.

categories were defined by the *Diff* values taken from four equally spaced time intervals over the 35 min PVT session, reflecting increasing levels of fatigue. Note that the categories are intended to be a rough guideline for decision making.

## 4 Conclusion and Outlook

Until now, using cognitive models for real-time monitoring has been hampered by practical limitations. These are the result of computationally intensive simulations required for parameter estimation, far exceeding the time available during task execution. Leveraging the PDLT method, we achieve real-time fatigue monitoring with the PVT, a proof-of-concept demonstration that performance can be remotely monitored with cognitive models in real time. We lay the foundation for real-time adaptive systems which rely on cognitive models to aid in recommending interventions designed to mitigate performance decrements by targeting appropriate cognitive mechanisms.

Our real-time fatigue monitoring paradigm can be expanded in several ways. First, we must integrate additional fatigue contributors like sleep history, consistent with fatigue risk management strategies [10–12]. This is particularly important for refining the ordinal fatigue categories. Parameter values derived from various levels of sleep deprivation should augment the time-on-task parameters for more robust categories reflecting the interaction of chronic sleep deprivation and acute fatigue. Additionally, we are currently extending the paradigm from performance monitoring to performance prediction. This entails a dynamic model in which key parameters are functions of time to capture increasing fatigue [15]. There are several benefits with this approach over the current moving window, such as stabilizing parameter estimates, eliminating correlated errors via conditional independence, and predicting performance decrements on future trials.

The PDLT method is agnostic to the specific model or application domain. Thus, if validated models are available, cognitive model-based monitoring can be implemented to capture and mitigate the effects of other moderators. For example, we might monitor task load impacts during dynamic multitasking, requiring multiple cognitive systems and multiple candidate sources of overload. Slow response speeds, for example, might be caused by missed information, sensory overload, or overtaxed memory retrieval. Internal states of the cognitive model will provide critical insights about the source of observed performance decrements. Values like number of items in a buffer or proportion of executed productions provide a novel set of metrics for the otherwise unobservable internal state of the operator. Thus, PDLT might be extended to store those internal states as well as the predicted behaviors for even deeper insights into cognitive states. Importantly, the PDLT, once established, can be accessed simultaneously for multiple observers, granting a single supervisor the ability to monitor multiple individuals simultaneously. Thus, the PDLT method brings us closer to a radically different future of cognitive model-based monitoring and performance enhancement.

**Acknowledgments.** We thank Brad Reynolds for software programming assistance. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This research was supported by a 711$^{th}$ Human Performance Wing Chief Scientist Seedling grant to G.G. and L.M.B.

## References

1. Anderson, J.R.: How Can the Human Mind Occur in the Physical Universe? Oxford University Press, New York (2007)
2. Basner, M., Dinges, D.F.: Maximizing sensitivity of the psychomotor vigilance test (PVT) to sleep loss. Sleep **34**(5), 581–591 (2011)
3. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computation (2014). arXiv preprint arXiv:1411.1607
4. Van den Bos, A.: Parameter Estimation for Scientists and Engineers. Wiley, New York (2007)
5. Bostock, M., Ogievetsky, V., Heer, J.: $D^3$ data-driven documents. IEEE Trans. Vis. Comput. Graph. **17**(12), 2301–2309 (2011)
6. Dinges, D.F., Powell, J.W.: Microcomputer analyses of performance on a portable, simple visual RT task during sustained operations. Behav. Res. Methods Instrum. Comput. **17**(6), 652–655 (1985)
7. Gluck, K.A., Gunzelmann, G.: Computational process modeling and cognitive stressors: background and prospects for application in cognitive. In: The Oxford Handbook of Cognitive Engineering, p. 424 (2013)
8. Gunzelmann, G., Gross, J.B., Gluck, K.A., Dinges, D.F.: Sleep deprivation and sustained attention performance: integrating mathematical and cognitive modeling. Cogn. Sci. **33**(5), 880–910 (2009)
9. Gunzelmann, G., Veksler, B.Z., Walsh, M.M., Gluck, K.A.: Understanding and predicting the cognitive effects of sleep loss through simulation. Trans. Issues Psychol. Sci. **1**(1), 106 (2015)
10. Hobbs, A., Avers, K.B., Hiles, J.J.: Fatigue risk management in aviation maintenance: current best practices and potential future countermeasures. Technical report, DTIC Document (2011)
11. Hursh, S.R., Redmond, D.P., Johnson, M.L., Thorne, D.R., Belenky, G., Balkin, T.J., Storm, W.F., Miller, J.C., Eddy, D.R.: Fatigue models for applied research in warfighting. Aviat. Space Environ. Med. **75**(Supplement 1), A44–A53 (2004)
12. Lerman, S.E., Eskin, E., Flower, D.J., George, E.C., Gerson, B., Hartenbaum, N., Hursh, S.R., Moore-Ede, M., et al.: Fatigue risk management in the workplace. J. Occup. Environ. Med. **54**(2), 231–258 (2012)
13. Loh, S., Lamond, N., Dorrian, J., Roach, G., Dawson, D.: The validity of psychomotor vigilance tasks of less than 10 min duration. Behav. Res. Methods **36**(2), 339–346 (2004)
14. Rangaiah, G.P.: Stochastic Global Optimization: Techniques and Applications in Chemical Engineering, vol. 2. World Scientific, Singapore (2010)
15. Veksler, B., Gunzelmann, G.: Functional equivalence of sleep loss and time on task effects in sustained attention (Under Review)
16. Walsh, M.M., Gunzelmann, G., Van Dongen, H.P.: Comparing accounts of psychomotor vigilance impairment due to sleep loss. In: Annual Meeting of the Cognitive Science Society, Pasadena, California, pp. 877–882 (2015)