Learning at Multiple Timescales

Matt Jones
Google Research, Brain Team
Mountain View, CA 94043
mcjones@google.com

Abstract

Natural environments have temporal structure at multiple timescales, a property that is reflected in biological learning and memory but typically not in machine learning systems. This paper advances a multiscale learning model in which each weight in a neural network is a sum of subweights learning independently at different timescales. A special case of this model is a fast-weights scheme, in which each original weight is augmented with a fast weight that rapidly learns and decays, enabling adaptation to distribution shifts during online learning. We then prove that more complicated models that assume coupling between timescales are equivalent to the multiscale learner, via a reparameterization that eliminates the coupling. Finally, we prove that momentum learning is equivalent to fast weights with a negative learning rate, offering a new perspective on how and when momentum is beneficial.

1 Introduction

Natural environments exhibit correlations at a wide range of timescales, a pattern variously referred to as self-similarity, power-law correlations, and $1/f$ noise [Kes82]. This is in stark contrast with the IID environments assumed by many machine learning (ML) methods. It also differs from diffusion or random-walk environments that are implicitly assumed by many online-learning methods such as temporal-difference learning and stochastic gradient descent (SGD), and which exhibit only short-range correlations. Moreover, biological learning systems are well-tuned to the temporal statistics of natural environments, as seen in phenomena of human cognition including power laws in learning [And82], power-law forgetting [WE97], long-range sequential effects [WJA$^+$13], and spacing effects [CVR$^+$08, AS91]. Thus an important goal for machine learning is to incorporate similar inductive biases into ML systems. This may yield improved performance in tasks with rich temporal structure, as arise in online or continual learning settings with real-world data.

This paper analyzes a simple and general framework for learning in temporally structured environments, *multiscale learning*, which in the context of neural networks (NNs) amounts to an optimizer. A common explanation for self-similar temporal structure in nature is that it arises from a mixture of events at various timescales. Indeed, many generative models of $1/f$ noise involve summing independent processes (e.g., wavelets or random walks) with varying time constants [EK09]. Accordingly, the multiscale optimizer assumes multiple learning processes operating in parallel at different timescales. In a NN, every weight is replaced by a family of subweights, each subweight having its own learning rate and decay rate, that sum to determine the weight as a whole. Learning at multiple timescales is a key idea in several theories in psychology and neuroscience, including conditioning [SCH02], learning [BF16], memory [MPC$^+$09, HK02], and motor control [KTS07]. The multiscale learner isolates and simplifies this idea, by assuming knowledge at different timescales evolves independently and that credit assignment follows standard gradient descent.

Here we prove exact correspondences between the multiscale optimizer and three other models: fast weights [cf. HP87, BHM$^+$16], the model synapse of Benna & Fusi [BF16], and momentum learning

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

[RHW86, Qia99]. The insight behind these proofs is that each of these models can be written in terms of a linear update rule with diagonalizable transition matrix. Thus the eigenvectors of this matrix all evolve independently. By writing the state of the model as a mixture of eigenvectors, we effect a coordinate transformation that exactly yields the multiscale optimizer. These results imply that the complicated coupling among timescales assumed by some models can be superfluous. More generally, they support the multiscale learner as a unifying framework for theories of learning at multiple timescales. Finally, they provide a new perspective on momentum learning, with implications for how and when it is beneficial.

2 Multiscale learner

Assume a statistical model $\hat{y}_t = h(\boldsymbol{x}_t, \boldsymbol{w}_t)$ and loss function $\mathcal{L}(y_t, \hat{y}_t)$, where $\boldsymbol{x}_t$ is the input on step t, $\boldsymbol{w}_t$ is the parameter estimate, $\hat{y}_t$ is the model output, and y_t is the target output. In a NN, $\boldsymbol{w}_t$ is the vector of current weights. For exposition, we assume the weights are updated by stochastic gradient descent (SGD), $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \nabla_{\boldsymbol{w}_t} \mathcal{L}(y_t, \hat{y}_t)$, and we henceforth abbreviate the gradient as $\partial_{\boldsymbol{w}_t} \mathcal{L}$. However, the following approach can be naturally composed with other optimizers, such as extensions of SGD or Hebbian learning, by replacing $-\alpha \partial_{\boldsymbol{w}_t} \mathcal{L}$ with the appropriate update term.

Multiscale learning is motivated by the assumption that, in online learning tasks, the true or optimal weights change over time, and that change occurs on multiple timescales. The multiscale optimizer expands each weight into a sum of subweights, $\boldsymbol{w} = \sum_{i=1}^{n} \boldsymbol{\omega}_i$, each with a different learning rate α_i and decay rate γ_i. Thus the subweights evolve according to:

$$\boldsymbol{\omega}_{i,t+1} = \gamma_i \boldsymbol{\omega}_{i,t} - \alpha_i \partial_{\boldsymbol{w}_t} \mathcal{L}. \tag{1}$$

Each $\boldsymbol{\omega}_i$ has characteristic timescale $\tau_i := (-\log \gamma_i)^{-1}$. Note that $\partial_{\boldsymbol{w}_t} \mathcal{L} = \partial_{\boldsymbol{\omega}_{i,t}} \mathcal{L}$, so one can think of the gradient for $\boldsymbol{w}$ being apportioned among the subweights (with total learning learning rate $\alpha = \sum \alpha_i$), or equivalently of each subweight following its own gradient.

3 Fast weights

An important special case of multiscale learning arises with two timescales, $\boldsymbol{w} = \boldsymbol{\omega}_{\text{slow}} + \boldsymbol{\omega}_{\text{fast}}$. We assume $\gamma_{\text{slow}} = 1$ (no decay) and $\alpha_{\text{fast}} > \alpha_{\text{slow}}$. Thus each component of $\boldsymbol{\omega}_{\text{slow}}$ can be thought of as the original weight, which is augmented by the corresponding component of $\boldsymbol{\omega}_{\text{fast}}$, a second channel between the same neurons that both learns and decays rapidly. The fast weights enable the system to adapt quickly to distribution shifts while resisting catastrophic forgetting (Figure 1).

This model is conceptually similar to the fast weights approach of Ba, Hinton and colleagues [HP87, BHM$^+$16] in that it implements a form of short-term memory in the weight matrix. In that previous work, the weights are updated by a different mechanism (Hebbian learning) than the primary weights, and they act as a memory of recent hidden states in a recurrent network. In the present conception, fast weights optimize the same loss as the primary weights, just with different temporal properties, and they act as a memory for recent learning signals (e.g., loss gradients). Thus they are perhaps better suited for handling distribution shifts of the sort considered here.

4 Benna-Fusi synapse

Benna and Fusi's model synapse [BF16] is designed to capture how biochemical mechanisms in real synapses implement a cascading hierarchy of timescales, and has been adopted in ML for continual reinforcement learning [KSC18, KSC19]. The model assumes that, within each individual weight w in a network, knowledge is maintained in a 1d hierarchy of variables $u_1, \ldots, u_n$, each dynamically coupled to its immediate neighbors in what is analogous to a flow equation:

$$C_1(u_{1,t+1} - u_{1,t}) = g_1(u_{2,t} - u_{1,t}) - \partial_{w_t} \mathcal{L} \tag{2}$$

$$C_k(u_{k,t+1} - u_{k,t}) = g_{k-1}(u_{k-1,t} - u_{k,t}) + g_k(u_{k+1,t} - u_{k,t}) \tag{3}$$

for $2 \leq k \leq n$, with $g_n = 0$. The external behavior of the synapse comes from u_1 alone (i.e., $w = u_1$), while $u_{2:n}$ act as stores with progressively longer timescales.

This update rule can be rewritten as $\boldsymbol{u}_{t+1} = \boldsymbol{T} \boldsymbol{u}_t - \boldsymbol{d}_t$, with transition matrix $\boldsymbol{T}$ determined by the coefficients in (2-3), and external signal $\boldsymbol{d}_t$ defined by $d_{1,t} = \frac{1}{C_1} \partial_{w_t} \mathcal{L}$ and $d_{2:n} \equiv 0$. It can be shown

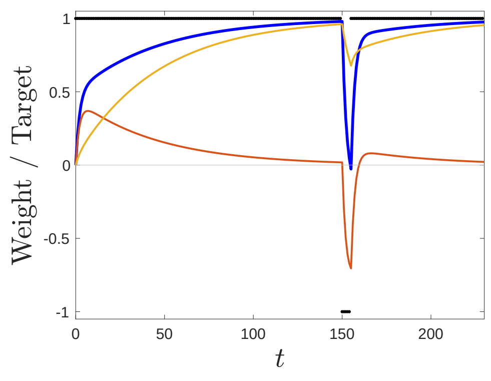

Figure 1: Illustration of fast weights. In this toy example, a single weight w (blue) with constant input ($x \equiv 1$) predicts a target signal T (black) with quadratic loss $\mathcal{L} = \frac{1}{2}(T - w)^2$. The weight is a sum of subweights w_{slow} (yellow) and w_{fast} (red). Initial learning is rapid, due to w_{fast}. Because of decay and the shared error signal, knowledge is gradually transferred to w_{slow} while w_{fast} returns to zero. When the task switches (trial 151), w_{fast} enables rapid adaptation while long-term knowledge is preserved in w_{slow}. Thus the model recovers quickly on the second reversal. The general multiscale learner extends this idea to an array of faster and slower weights.

that the transition matrix is diagonalizable, $\boldsymbol{T} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$, with eigenvalues $\Lambda_{ii} = \lambda_i < 1$ (see Appendix). We can further enforce $\boldsymbol{V}_{1\cdot} = 1$, for a reason given below. We refer to the eigenvectors (columns $\boldsymbol{V}_{\cdot i}$) as *modes* of the system, because they are preserved over time up to a scalar. That is, if the initial state is proportional to mode i, then in the absence of external signal ($\boldsymbol{d} \equiv \boldsymbol{0}$), the system will remain in that mode, decaying exponentially with rate factor λ_i:

$$\boldsymbol{u}_0 \propto \boldsymbol{V}_{\cdot i} \quad \Longrightarrow \quad \forall_t : \boldsymbol{u}_t = \lambda_i^t \boldsymbol{u}_0. \tag{4}$$

In general, any state can be written uniquely as a linear combination of modes, $\boldsymbol{u} = \sum \omega_i \boldsymbol{V}_{\cdot i} = \boldsymbol{V}\boldsymbol{\omega}$. Therefore we can reparameterize the model as $\boldsymbol{\omega} := \boldsymbol{V}^{-1}\boldsymbol{u}$, leading to the simplified update equation:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\Lambda}\boldsymbol{\omega}_t + \boldsymbol{V}^{-1}\boldsymbol{d}_t. \tag{5}$$

Because $\boldsymbol{\Lambda}$ is diagonal, there is no cross-talk between the modes, unlike in the original dynamics. Thus we have derived the multiscale learner, with subweights $\omega_{i,t}$, decay rates λ_i, and learning rates $\frac{1}{C_1}\left[\boldsymbol{V}^{-1}\right]_{i1}$. The assumption $\boldsymbol{V}_{1\cdot} = 1$ implies $u_1 = \sum \omega_i$, so the models agree on the external behavior of the weight as a whole. Figure 2 illustrates the translation between the two models.

5 Momentum learning

The standard rationale for momentum learning is to smooth updates over time, so that oscillations along directions of high curvature cancel out while progress can be made in directions with consistent gradients [RHW86]. The momentum is defined as an exponentially filtered running average of gradients, with weight update determined by current momentum:

$$\boldsymbol{g}_{t+1} = \beta \boldsymbol{g}_t + (1-\beta)\partial_{\boldsymbol{w}_t}\mathcal{L} \tag{6}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}_{t+1}. \tag{7}$$

This formulation is equivalent to one in which the update $\Delta\boldsymbol{w}_t = \boldsymbol{w}_{t+1} - \boldsymbol{w}_t$ includes a portion of the previous update: $\Delta\boldsymbol{w}_t = -\alpha\partial_{\boldsymbol{w}_t}\mathcal{L} + \beta\Delta\boldsymbol{w}_{t-1}$, with $\alpha = \eta(1-\beta)$.

Paralleling the analysis in Section 4, and again focusing on a single weight in the network to simplify notation, we write the state of the momentum optimizer as $[w, g]^\top$ and use (6-7) to obtain the update rule:

$$\begin{bmatrix} w \\ g \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & -\eta\beta \\ 0 & \beta \end{bmatrix} \begin{bmatrix} w \\ g \end{bmatrix}_t + \begin{bmatrix} -\eta(1-\beta) \\ (1-\beta) \end{bmatrix} \partial_{w_t}\mathcal{L}. \tag{8}$$

The transition matrix in (8) has eigenvectors $[1, 0]^\top$ with eigenvalue 1, and $[1, \frac{1-\beta}{\eta\beta}]^\top$ with eigenvalue β. Now use this eigenbasis to define a reparameterization:

$$\begin{bmatrix} w \\ g \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & \frac{1-\beta}{\eta\beta} \end{bmatrix} \begin{bmatrix} \omega_{\text{slow}} \\ \omega_{\text{fast}} \end{bmatrix}. \tag{9}$$

Substitution into (8) yields the reparameterized update rule:

$$\begin{bmatrix} \omega_{\text{slow}} \\ \omega_{\text{fast}} \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} \omega_{\text{slow}} \\ \omega_{\text{fast}} \end{bmatrix}_t - \begin{bmatrix} \eta \\ -\eta\beta \end{bmatrix} \partial_{w_t}\mathcal{L}. \tag{10}$$

Thus we have recovered the fast-weight model, with decay $\gamma_{\text{fast}} = \beta$ and learning rates $\alpha_{\text{slow}} = \eta$ and $\alpha_{\text{fast}} = -\eta\beta$. The negative fast learning rate is perhaps surprising and is discussed in Section 6.

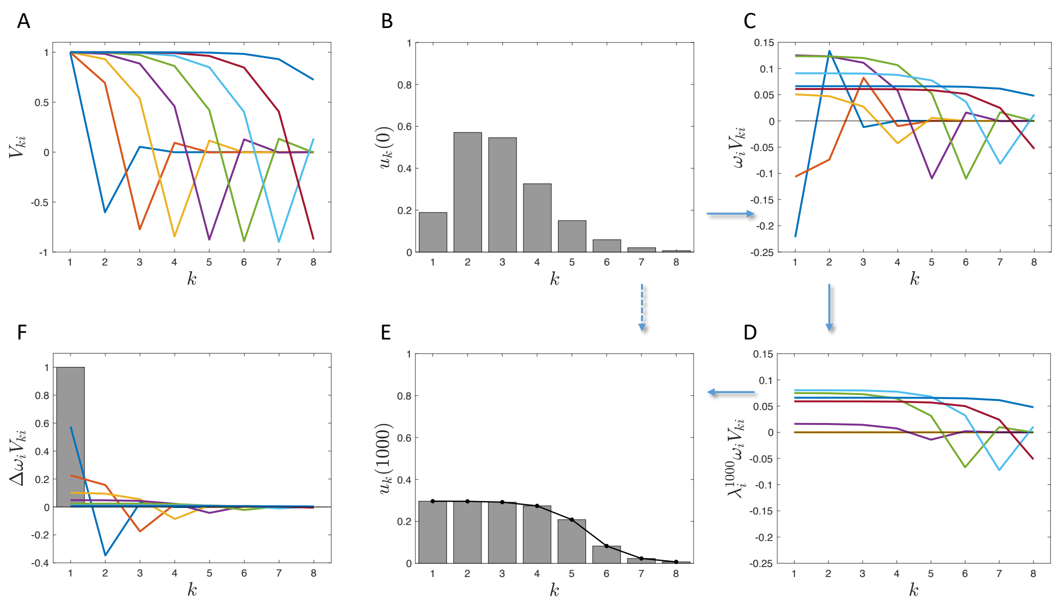

Figure 2: Translation between Benna-Fusi model [BF16] and multiscale optimizer works by decomposing the state of the former into modes, or eigen-patterns of activation that decay independently, which correspond to subweights in the multiscale optimizer. A: All modes for a default Benna-Fusi model with $n = 8$ variables. B: An arbitrary initial state. C: Unique eigen-decomposition of the state in Fig 2B. Implied values of the multiscale optimizer's subweights can be read off as the values of the curves at $k = 1$. D: Decay of the individual modes or subweights for 1000 steps (with no external input) at rates given by their eigenvalues. E: Reconstruction of the final state exactly matches the result of iterating the Benna-Fusi update (dotted arrow from Fig 2B). F: Decomposition of a unit impulse to u_1 (e.g., loss gradient, shown as grey bar) as a weighted sum of modes. Learning rates for the corresponding subweights can be read off as the values of the curves at $k = 1$ (because $\boldsymbol{V}_{1i} = 1$).

6 Conclusions

The equivalences proved here, and others provable by similar methods, simplify the space of possible models and provide a unified framework for investigating learning at multiple timescales. A key feature of the multiscale optimizer is that the components all evolve independently, both in weight decay between updates and in updating from their individual loss gradients. This makes the model both conceptually and computationally simpler than past ones. Simulations reported in a companion paper, using an online prediction task governed by $1/f$ noise in latent environmental parameters, show that the multiscale optimizer significantly outperforms SGD and batch learning with finite memory horizon, and nearly matches Bayes-optimal performance without the computational overhead [JSE$^+$22]. Taking all of these results together, the multiscale optimizer enjoys a combination of normative, heuristic, and biological justification, good performance, and computational efficiency. Future work will incorporate this method into state-of-the-art deep NN architectures, to explore whether it yields ML systems that are better able to exploit temporal structure in natural environments.

A further contribution is the characterization of momentum as a fast weight with negative learning rate. Although this connection may seem counterintuitive, it can be understood as follows: When $\alpha_{\text{fast}} < 0$, the subweights learn in opposite directions, with the latent knowledge in ω_{slow} overshooting the overt knowledge in $w = \omega_{\text{slow}} + \omega_{\text{fast}}$. As ω_{fast} decays upward toward 0, w catches up to ω_{slow}, and the model appears to continue learning from past input. Momentum acts in the same manner, as a form of latent knowledge of past input that continues to drive later updates. This correspondence is exact under the translation in Section 5. Put differently, learning at multiple timescales is motivated by an expectation of positive autocorrelation in the environment, whereas momentum is effective at smoothing out negative autocorrelation in the gradient signal (e.g., along highly curved directions in weight space). The inherent opposition between these considerations may be reconcilable using the present results that place momentum learning within the broader framework of multiscale learning.

References

[And82] John R. Anderson. Acquisition of cognitive skill. *Psychological Review*, 89:369–406, 1982.

[AS91] John R. Anderson and Lael J. Schooler. Reflections of the environment in memory. *Psychological Science*, 2(6):396–408, 1991.

[BF16] Marcus K. Benna and Stefano Fusi. Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19, 2016.

[BHM$^+$16] Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.

[CVR$^+$08] Nicholas J. Cepeda, Edward Vul, Doug Rohrer, John T. Wixted, and Harold Pashler. Spacing effects in learning: A temporal ridgeline of optimal retention. *Psychological Science*, 19(11):1095–1102, 2008.

[EK09] Iddo Eliazar and Joseph Klafter. A unified and universal explanation for lévy laws and 1/f noises. *Proceedings of the National Academy of Sciences*, 106(30):12251–12254, 2009.

[HK02] Marc W. Howard and Michael J. Kahana. A distributed representation of temporal context. *Journal of mathematical psychology*, 46(3):269–299, 2002.

[HP87] Geoffrey E. Hinton and David C. Plaut. Using fast weights to deblur old memories. In *Proceedings of the 9th annual conference of the cognitive science society*, pages 177–186, 1987.

[JSE$^+$22] Matt Jones, Tyler Scott, Gamaleldin ElSayed, Mengye Ren, Katherine Hermann, David Mayo, and Michael C. Mozer. Neural network online training with sensitivity to multiscale temporal structure. In *NeurIPS workshop on Memory in Artificial and Real Intelligence (MemARI)*, 2022.

[Kes82] Marvin S. Keshner. 1/f noise. *Proceedings of the IEEE*, 70(3):212–218, 1982.

[KSC18] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018. PMLR 80.

[KSC19] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, 2019. PMLR 97.

[KTS07] Konrad P. Kording, Joshua B. Tenenbaum, and Reza Shadmehr. The dynamics of memory as a consequence of optimal adaptation to a changing body. *Nature Neuroscience*, 10(6):779–786, 2007.

[MPC$^+$09] Michael C. Mozer, Harold Pashler, Nicholas Cepeda, Robert V. Lindsey, and Ed Vul. Predicting the optimal spacing of study: A multiscale context model of memory. *Advances in neural information processing systems*, 22, 2009.

[Qia99] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel distributed processing, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[SCH02] JER Staddon, IM Chelaru, and JJ Higa. Habituation, memory and the brain: The dynamics of interval timing. *Behavioural Processes*, 57(2-3):71–88, 2002.

[WE97] John T. Wixted and Ebbe B. Ebbesen. Genuine power curves in forgetting: A quantitative analysis of individual subject forgetting functions. *Memory & Cognition*, 25(5):731–739, 1997.

[WJA$^+$13] Matthew H. Wilder, Matt Jones, Alaa A. Ahmed, Tim Curran, and Michael C. Mozer. The persistent impact of incidental experience. *Psychonomic bulletin & review*, 20(6):1221–1231, 2013.

A Appendix

This appendix provides details on diagonalizing the transition matrix of the update rule for the Benna-Fusi model, and the corresponding reparameterization in terms of the eigenbasis.

First, reparameterize the Benna-Fusi model so that its transition matrix is symmetric, as follows. Recursively define

$$b_k = \begin{cases} 1 & k = 1 \\ \frac{b_{k-1} c_k}{c_{k-1}} & 1 < k \leq n \end{cases} \tag{11}$$

and write the state of the Benna-Fusi synapse as

$$\boldsymbol{\phi} = (\sqrt{b_k} u_k)_{1 \leq k \leq n}. \tag{12}$$

The update becomes

$$\boldsymbol{\phi}_{t+1} = \boldsymbol{\Gamma} \boldsymbol{\phi}_t + \boldsymbol{d}_t \tag{13}$$

with

$$\Gamma_{k,k} = 1 - \frac{g_{k-1} + g_k}{C_k} \tag{14}$$

$$\Gamma_{k-1,k} = \Gamma_{k,k-1} = \frac{g_{k-1}}{\sqrt{C_{k-1} C_k}}. \tag{15}$$

Symmetry of $\boldsymbol{\Gamma}$ implies it has an orthonormal eigenbasis, $\{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n\}$, with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$. Because the scaling of eigenvectors is arbitrary, we can enforce $\psi_{i,1} = 1$ for all i.

To translate the eigenbasis back to $\boldsymbol{u}$, define $\boldsymbol{B} = \mathrm{diag}(\sqrt{\boldsymbol{b}})$ so that $\boldsymbol{\phi} = \boldsymbol{B}\boldsymbol{u}$ and $\boldsymbol{T} = \boldsymbol{B}^{-1} \boldsymbol{\Gamma} \boldsymbol{B}$. It is then easily verified that

$$\boldsymbol{V}_{\cdot i} = \boldsymbol{B}^{-1} \boldsymbol{\psi}_i, \tag{16}$$

is an eigenvector of $\boldsymbol{T}$ with eigenvalue λ_i. Therefore $\boldsymbol{T} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^{-1}$, as claimed in the main text. Note that the choice $\psi_{i,1} = 1$ entails $V_{1,i} = 1$ (because $b_1 = 1$). Finally, $\boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n]$ is invertible because it comprises a basis, and therefore so is $\boldsymbol{V} = \boldsymbol{B}^{-1} \boldsymbol{\Psi}$. Therefore the reparameterization $\boldsymbol{u} \mapsto \boldsymbol{\omega} = \boldsymbol{V}^{-1} \boldsymbol{u}$ is well-defined.