SIAM J. Applied Dynamical Systems
Vol. 18, No. 3, pp. 1293–1333

The Dynamics of Swamps in the Canonical Tensor Approximation Problem*

Martin J. Mohlenkamp†

Abstract. The ability to approximate a multivariate function/tensor as a sum of separable functions/tensors is quite useful. Unfortunately, optimization-based algorithms to do so regularly exhibit unusual transient behavior, known informally as transient swamps, when the error decreases by minuscule amounts for many iterations but then decreases more rapidly. Such swamps have been the bane of users for decades and their cause a mystery. Here we analyze a model problem (that of attempting to recover a rank-2 tensor) and the dynamics of some example swamps that occur within it. We find that transient swamps are caused by saddle-like essential discontinuities, which we dub essential saddles. Descent from an essential saddle is constrained within a steep-sided valley, which causes algorithms to zig-zag and progress slowly. Essential saddles are a surprising mechanism and are not amenable to standard analysis, which explains why they have been so hard to identify.

Key words. tensor approximation, alternating least-squares (ALS), canonical tensor format, swamps, essential saddle

AMS subject classifications. 15A69, 37N30, 65K10

DOI. 10.1137/18M1181389

1. Introduction.

Consider the problem of approximating a given target tensor T by another tensor G that is written as a (short) sum of separable tensors, which we can write as

$$T \approx G = \sum_{l=1}^{r} \bigotimes_{i=1}^{d} G_i^l \,. \tag{1.1}$$

This sum-of-separable format for G is known variously as the canonical format, the canonical polyadic format, the CANDECOMP/PARAFAC format, or a separated representation and has several important applications. There has been steady, but slow, progress in understanding aspects of the approximation problem (1.1). However, our understanding in the true tensor case of $d > 2$ is far more primitive than our understanding in the matrix case $d = 2$. We will assume throughout this work that $d > 2$.

It is NP-hard to find G with given r that minimizes the error in (1.1) or to solve most other tensor problems. Optimization-based algorithms have been developed based on several different optimization strategies, such as alternating least-squares (ALS), Newton's method, nonlinear conjugate gradient, and line search. In many cases these algorithms can quickly

*Received by the editors April 19, 2018; accepted for publication (in revised form) by M. Pugh May 28, 2019; published electronically July 23, 2019.
https://doi.org/10.1137/18M1181389
Funding: This research is based upon work supported by the National Science Foundation under grant 1418787.
†Department of Mathematics, Ohio University, Athens, OH 45701 (mohlenka@ohio.edu).

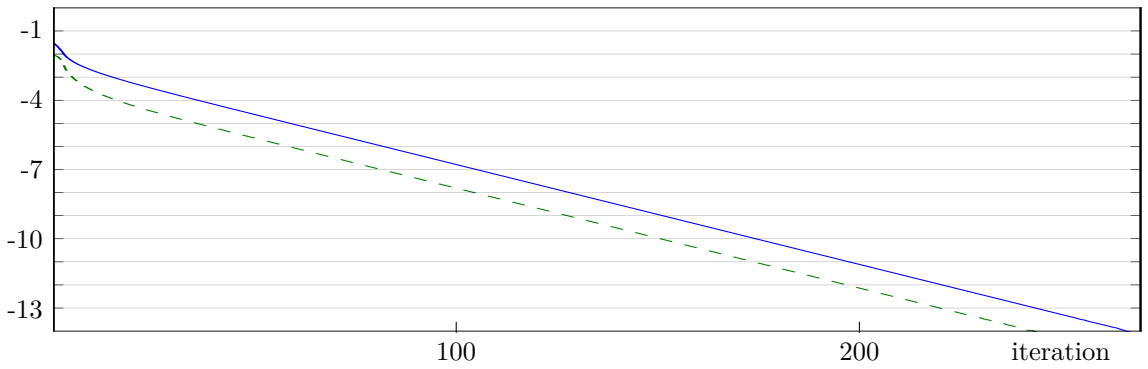

Figure 1.1. *Illustration of a terminal swamp. The top (blue) curve is* $\log_{10}$ *of the error and the bottom (green) curve is* $\log_{10}$ *of the difference in error of consecutive iterations with vertical axis* $[-14, 0]$*. The horizontal axis is iteration number in* $[0, 270]$*.*

produce a good-quality G. However, in other cases they are prone to poor behavior, such as long periods in which the iteration reduces the error by miniscule amounts, which are known informally as *swamps*. Work has been done trying to understand and alleviate swamps, but swamps remain a significant impediment to the use of methods based on (1.1). We will distinguish two types of swamps:

- A *terminal swamp* is when the local convergence to a (local) minimum is slow, either by being sublinear or linear with a poor rate constant. A typical error plot in the linear case is shown in Figure 1.1. Such behavior occurs in many optimization problems and is generally associated with a large condition number in the Hessian of the error function at the minimum. Thus the direct cause is relatively well understood and the question becomes why and when such ill-conditioned Hessians arise in the tensor approximation problem (1.1).
- A *transient swamp* is when the error decreases by minuscule amounts for many iterations but then decreases more rapidly. In Figure 1.2 we show such a transient swamp, produced using the ALS algorithm. It takes only two iterations to reduce the error from nearly 1 to just above 10^{-2}, 597 more iterations to reduce the error below 10^{-2}, and then 10 more iterations to reduce it below 10^{-14}. The cause of transient swamps has been a great mystery. Modifications to ALS and alternatives to ALS have been developed to try to cure transient swamps but without an understanding of what causes transient swamps.

In [17] we started a project to better understand the approximation problem (1.1) with focus on the causes of swamps. We refer the reader there for more extensive background discussion and references to the literature.

We consider (1.1) as a problem of minimizing an objective function $f(\mathbf{x})$, where $\mathbf{x}$ parameterizes G. We restrict ourselves to least-squares approximation, and so $f(\mathbf{x}) = \|T - G\|^2$ using the Frobenius norm with an optional additional regularization term $\lambda \sum_{l=1}^{r} \| \bigotimes_{i=1}^{d} G_i^l \|^2$. We then consider the gradient flow $\dot{\mathbf{x}} = -\nabla f(\mathbf{x})$ as an attempt to minimize $f(\mathbf{x})$, and we analyze (1.1) using methods of dynamical systems.

Although the gradient flow provides useful information about the behavior of algorithms,

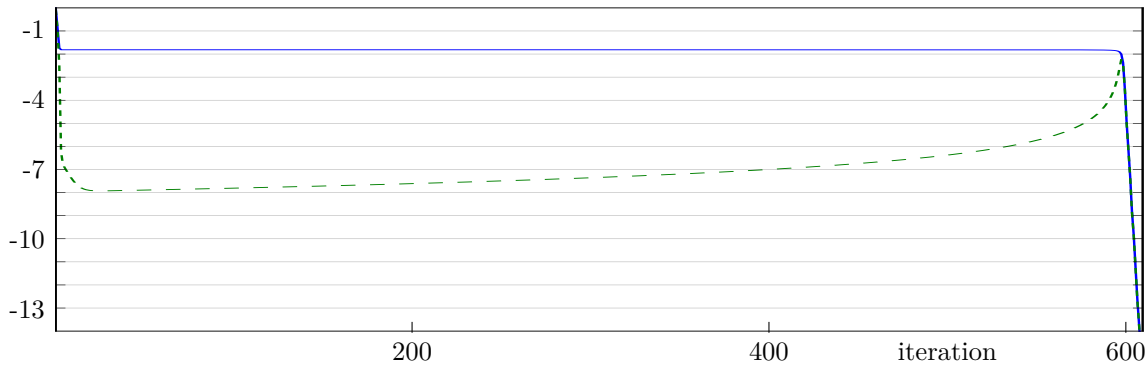

Figure 1.2. *Illustration of a transient swamp in the same format as Figure 1.1 but with iteration number in* $[0, 610]$.

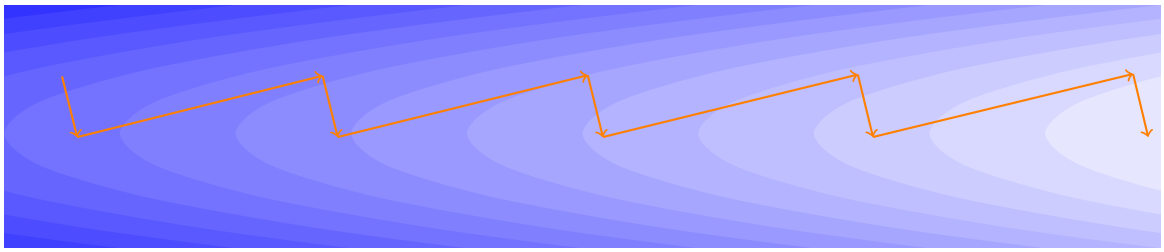

Figure 1.3. *Behavior of gradient descent with line search in a valley, illustrating the zig-zag behavior caused by the valley sides.*

one soon realizes there is a mismatch. While small $\|\nabla f(\mathbf{x})\|$ makes the flow slow, it does not necessarily make algorithms progress slowly. Even the simple algorithm of gradient descent with line search may still take a large step if $f(\mathbf{x})$ keeps descending in the direction of $-\nabla f(\mathbf{x})$. Instead, the (local) progress rate of algorithms also depends on what happens transverse (i.e., orthogonal) to the gradient. If the transverse Hessian has only positive eigenvalues, then the algorithm will in general take a zig-zag path, as illustrated in Figure 1.3 in a valley. In [17] we determined that the algorithm progress rate is governed by the ratio of $\|\nabla f(\mathbf{x})\|$ to the largest eigenvalue of the Hessian (assuming all the eigenvalues are positive). The first contribution of [17] was to develop an analysis framework for understanding (1.1) using methods of dynamical systems, including the identification of key quantities such as this ratio. In section 2 we briefly review this framework.

The second contribution of [17] was to start the analysis of a model problem. We will only consider real tensors and approximations over the reals. The simplest nontrivial case for T is when it has rank two and thus can be written as $T = \sum_{l=1}^{2} \bigotimes_{i=1}^{d} T_i^l$. It can then be put into a standard form

$$T = \left(1 + 2z \prod_{i=1}^{d} \cos(\phi_i) + z^2\right)^{-1/2} \left(\bigotimes_{i=1}^{d} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + z \bigotimes_{i=1}^{d} \begin{bmatrix} \cos(\phi_i) \\ \sin(\phi_i) \end{bmatrix}\right), \tag{1.2}$$

where $|z| \leq 1$, $\phi_i \in [0, \pi/2]$, and the leading scalar ensures $\|T\| = 1$. We thus have target

tensors parameterized by d, z, and the vector of angles $\boldsymbol{\phi}$. For G we can then consider G_1 with too small rank $r = 1$, G_2 with the correct rank $r = 2$, and G_3 with excess rank $r = 3$. Under the additional assumption that $G_i^l \in \text{span}\{T_i^1, T_i^2\}$, the transformation that put T in the form (1.2) puts G_1 in the form

$$G_1 = a \bigotimes_{i=1}^{d} \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix} . \tag{1.3}$$

In [17] we analyzed the case of $G_1 = (1.3)$ approximating $T = (1.2)$. We found the following:

- As the parameters z and $\boldsymbol{\phi}$ change, there are bifurcations in the qualitative behavior of the approximation problem.
- Terminal swamps can occur near these bifurcation points when the minimum becomes nonhyperbolic.
- Transient swamps can occur for certain parameter values when saddles become nonhyperbolic.

In the current work, we consider approximating $T = (1.2)$ with

$$G_2 = a \bigotimes_{i=1}^{d} \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix} + b \bigotimes_{i=1}^{d} \begin{bmatrix} \cos(\beta_i) \\ \sin(\beta_i) \end{bmatrix} . \tag{1.4}$$

There are two main aspects in which approximating by G_2 differs from approximating by G_1. First, we can achieve equality $G_2 = T$. Thus we can discuss and analyze the behavior at "solutions," rather than just minima. (We could then talk of "recovering" T, but our focus is on the approximation process.) Second, there are cases where the gradient flow reduces the error monotonically and has convergent angles $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ but has divergent scalars $(|a|, |b|) \to (\infty, \infty)$; we will call these *local infima*. The phenomenon of local infima is closely related to the ill-posedness of approximating a general tensor with a rank-2 tensor and to rank-jumping limits (see, e.g., [9, 11, 23, 41, 42, 43, 44, 45, 46, 47, 50]). Rank-jumping limits can be partially understood by observing that $T = (1.2)$ is undefined when $(z, \boldsymbol{\phi}) = (-1, \mathbf{0})$ due to division by zero. If we fix a ray along which $(z, \boldsymbol{\phi}) \to (-1, \mathbf{0})$, then the limit of T exists (and is computable using L'Hôpital's rule) and can have rank up to d. Similarly, $G_2/\|G_2\|$ is undefined when $(-b, \boldsymbol{\beta}) = (a, \boldsymbol{\alpha})$, but the limit as $(-b, \boldsymbol{\beta}) \to (a, \boldsymbol{\alpha})$ exists along rays. In both cases, the value of the limit depends on the ray.

There are three perspectives from which we consider the approximation of T by G_2:

1. We obtain analytic results with proofs for some specific cases.
2. We plot various quantities from the framework developed in [17] to develop understanding.
3. We numerically generate example swamps and analyze them.

These three perspectives feed off each other: analytic results suggest plots to look at and parameters that might generate swamps, plots suggest further parameters that might generate swamps, swamp examples suggest symmetries in which to plot and look for analytic results, etc. As a result, there is not a linear, logical ordering for the content of this work. Since our goal is to understand swamps, and the model problem of the approximation of T by G_2 is only a means to that goal, we will focus on item 3. In section 3 we sketch the analytic work for

item 1, but we relegate the formulas and proofs to Appendix A. In section 3 we also describe the plots available for item 2 and present those plots for a single choice of parameters. In the supplementary material (M118138_01.pdf [local/web 33.5MB]), linked from the main article webpage, we present plots for a variety of parameter choices.

In section 4 we address item 3 by generating several example swamps and analyzing their dynamics using the tools in section 3. These numerical examples are all generated using the ALS algorithm because it is well known and the manifestation of swamps is clearest within it (see, e.g., [10, 25, 26, 28, 31, 34, 36, 37]). We believe that our analysis yields compelling explanations for these swamps and that features in the flow path that we identify are the *causes* of the swamps. However, we do implicitly assume that the orbit produced by the ALS algorithm qualitatively matches the gradient flow, which will not always be the case. Thus we produce explanations but no rigorous statements. Similarly, we believe the flow features that we have identified are important for other algorithms but have no rigorous statement to that effect. These swamps seem generic enough to serve as prototypes, but there may be swamps with other causes.

First, in subsection 4.1 we analyze the terminal swamp shown in Figure 1.1. It is produced choosing $d = 6$, $z = 1$, and all $\phi_i = \phi = \pi/8$. As expected, it is due to an ill-conditioned Hessian at the solution. We can compute the condition number explicitly and prove that the condition number goes to infinity as $\phi \to 0^+$. This behavior agrees with observations in the literature that slow convergence correlates with small angles (called ill-conditioning or degeneracy) (see, e.g., [10, 26, 28, 31, 36, 37]). Qualitatively, this swamp occurs because one of the two terms in G_2 quickly moves near the best G_1 approximation (with $\alpha_i \approx \phi/2$) while the second term does not help reduce the error appreciably (with $\beta_i \approx 2\phi$). Then, very slowly, $\alpha_i \to 0^+$ and $\beta_i \to \phi^+$ for all i, which gives the solution.

Second, in subsection 4.2 we analyze the transient swamp shown in Figure 1.2. It is produced choosing $d = 6$, $z = 1$, and all $\phi_i = \phi = \pi/8$, which are the same parameters as the terminal swamp above; only the initial $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is different. In the first few iterations, the error reduces significantly and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ change rapidly. Qualitatively, *both* terms in G_2 move toward the G_1 solution, resulting in $\alpha_i \approx \beta_i \approx \phi/2$ for all i. To escape from this state, $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ break symmetry, with one direction (say $i = 1$) behaving differently from the others, and move to $\alpha_1 \approx \beta_1 \approx \phi/2 \pm \pi/2$ and $\alpha_i \approx \beta_i \approx \phi/2$ for all $i > 1$. Overall, the action is to funnel some volume in $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-space toward a specific point, which is the origin of the swamp.

The variables $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ are then very close to a distinctive feature in the error function. As a simplified model of this feature, consider the function $f(x, y)$ expressed in polar coordinates as $f(x, y) = f(r\cos(\theta), r\sin(\theta)) = -(1 + r^2)\cos(2\theta)$ and illustrated in Figure 1.4. The value $f(0, 0)$ is undefined, and the limit $\lim_{(x,y)\to(0,0)} f(x, y)$ does not exist. However, for any fixed θ, $\lim_{r\to 0} f(x, y) = -\cos(2\theta)$ exists. Thus f has a discontinuity at $(0, 0)$ and this discontinuity is not one of the named types (e.g., removable), and so it is "essential." The gradient flow due to f moves towards $(0, 0)$ along $\theta = \pm\pi/2$ and away from $(0, 0)$ along $\theta = 0$ and π, and so $(0, 0)$ behaves like a saddle. We will call this saddle-like essential discontinuity an *essential saddle*. In the descent directions $\theta = 0$ and π, f has a steep-sided valley with steepness (measured by the largest eigenvalue of the transverse Hessian) going to infinity as $r \to 0^+$. Thus algorithms will zig-zag with very small steps (as in Figure 1.3) but with steps slowly growing. Returning now to the specific example in subsection 4.2, we find that descent from such an essential

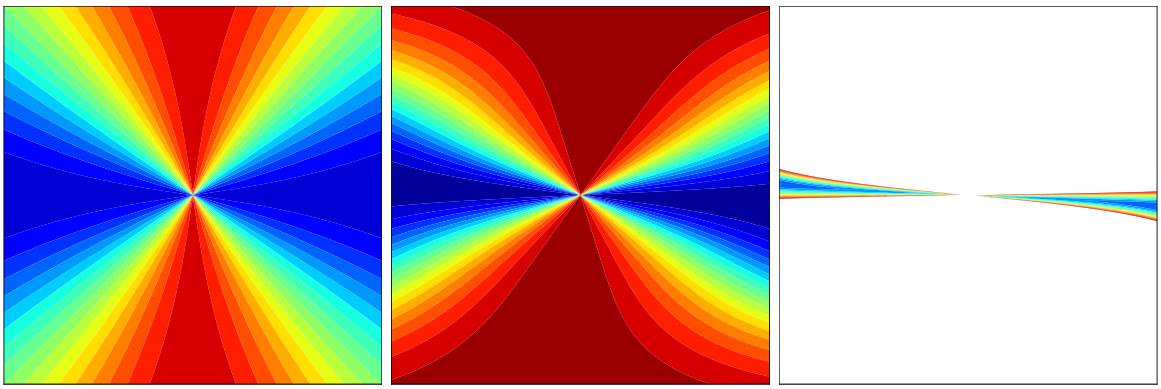

Figure 1.4. *Illustration of essential saddles using contour plots with blue small and red large. The left panel is the function* $f(x,y) = f(r\cos(\theta), r\sin(\theta)) = -(1+r^2)\cos(2\theta)$ *on* $(x,y) \in [-1/4, 1/4]^2$ *with contours in* $[-1.2, 1.2]$*. The center panel is the error function that appears in the transient swamp in subsection 4.2 with contours in* $[0,1]$*. The right panel is the same error function but with contours in* $[0.01, 0.03]$ *and white indicating values above* 0.03*. The right panel shows that the descent direction is in an extremely steep-sided valley.*

saddle is the *direct cause* of the transient swamp. Eventually $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ leaves the vicinity of the essential saddle and then converges rapidly to the solution (along a subspace in which the Hessian is well-conditioned). The essential saddle is robust with respect to perturbations in ϕ and worsens as $\phi \to 0^+$.

Third, in subsection 4.3 we consider a swamp that has a transient phase followed by a terminal phase. It is produced choosing $d = 6$, $z = -1$, and all $\phi_i = \phi = \pi/8$. As with the example in subsection 4.2, the transient swamp is caused by an essential saddle, and as with the example in subsection 4.1, the terminal swamp is caused by an ill-conditioned Hessian at the solution. Both the transient and the terminal parts are robust with respect to perturbations in ϕ and worsen as $\phi \to 0^+$. Qualitatively, this swamp can be well explained using various plots and symmetries but lacks interpretation in terms of the best G_1 approximation.

Fourth, in subsection 4.4 we consider in detail a swamp with $d = 3$ used by Paatero in [34]. In our interpretation, $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is first funneled toward an essential saddle, then has a short transient swamp that leads to another essential saddle, then has a long transient swamp, and finally has a long terminal swamp due to an ill-conditioned Hessian. Paatero says the long transient swamp and long terminal swamp together are caused by the tensor having acquired degeneracy during the earlier phases. Here degeneracy means being near a higher rank tensor that can be achieved as a rank-jumping limit, and it is characterized by large values of $(|a|, |b|)$ and hence large cancellation in (1.4). We find that degeneracy is a *symptom* of a transient swamp, whereas the essential saddle is its *cause*. Similarly, ill-conditioning of the linear system in ALS (as observed in [31, 37]) and large values of the tensor condition number of [4, 5, 6, 7] are symptoms rather than causes.

Fifth, in subsection 4.5 we briefly discuss some other swamps. In particular we find that the swamps for G_1 identified in [17] are quite sensitive to the choice of parameters, which leads us to conclude they are unlikely to be observed in practice and thus are *not* the swamps observed in the literature.

Our main conclusions and claims are therefore the following:

- As expected, terminal swamps are caused by an ill-conditioned Hessian.
- Transient swamps are primarily caused by an unexpected feature, an essential saddle.
- As expected, both types worsen as the angle between terms becomes smaller.

We have shown that essential saddles cause transient swamps in simple rank-2 model problems and that they are robust to perturbations. In considering higher-rank approximation problems, one should first note that they can contain embedded rank-2 approximation problems and thus can have essential saddles and the transient swamps caused by them. Second, one would suppose that nontrivial behaviors caused by a subset of the terms would be more common than nontrivial behaviors requiring the interaction of all the terms. Thus, while more complex causes for transient swamps will likely also occur, we believe that essential saddles are a common cause of the observed transient swamps and we conjecture that they are the primary cause.

The ill-conditioned Hessians and essential saddles that we find cause swamps are features of the approximation problem (1.1) itself, not of any particular algorithm. However, different algorithms will be affected by them in different ways and to different extents. The ill-conditioned Hessians causing terminal swamps are conventional, and many non-ALS algorithms have good theoretical basis for claiming superiority over ALS; variations on ALS may also be effective. The clearest manifestations of transient swamps are within ALS (see, e.g., [10, 25, 26, 28, 31, 34, 36, 37]). Proposers of non-ALS methods (see, e.g., [1, 6, 7, 13, 15, 16, 22, 33, 35, 40, 48]) generally avoid saying that transient swamps still occur but do not make claims that they do not. Numerical studies comparing the proposed algorithm to existing algorithms focus on average performance and thus do not address the issue. Although the numerical tests always show the superiority of the proposed method (otherwise it would not be published), there are always lingering questions on the appropriateness of the test suite and, ultimately, on the *cause* of the observed performance; consequently, the old, simple, ALS algorithm remains popular. We cannot fault the proposers of other algorithms for this situation: since the cause of transient swamps was not known, they had no possibility to show their algorithm performed better with regard to transient swamps.

Now that essential saddles have been identified as the cause of transient swamps, the available algorithms should be re-evaluated. However, a thorough evaluation is beyond the scope of this work. In section 5 we sketch general considerations for converging in terminal swamps, escaping from transient swamps, and avoiding entering transient swamps. We then consider a simple variation of ALS where regularization is temporarily turned on and show that it may be used to avoid or escape from transient swamps. Now that the cause of transient swamps is identified, producing an effective algorithm is not so daunting. Existing algorithms, with small modifications and a touch of adaptivity, may be sufficient.

Finally, we note some implications for the field of dynamical systems. We have shown that essential saddles occur naturally in an important application, which motivates their further study. The eigenvalues of the Hessian are the key to the analysis of ordinary saddles, but the Hessian does not exist at an essential saddle. Moreover, the illustration in Figure 1.4, which shows the essential saddle as a point in a two-dimensional space, is misleadingly simple. In the example of subsection 4.2, there is a discontinuity along a six-dimensional set in a 12-dimensional space. The point we identify as the essential saddle appears to have 10

strongly attracting directions, one weakly attracting direction, and one weakly repelling direction. Some of the strongly attracting directions, which would normally be tangent to the stable manifold, may instead be tangent to the six-dimensional set of discontinuities, further complicating the analysis. How does one analyze and compare essential saddles?

2. Analysis framework.

In this section we briefly review the analysis framework from [17] so that we have the notation and terminology at hand.

Suppose we have a differentiable objective function $f(\mathbf{x})$ defined on some connected domain. By the *gradient flow* of f we mean the flow induced by the differential equation

$$\dot{\mathbf{x}}(t) = -\nabla f(\mathbf{x}(t)). \tag{2.1}$$

An *orbit* is a realization of the flow from a particular starting point $\mathbf{x}(0) = \mathbf{x}_0$. A *stationary point* is a point $\mathbf{x}$ where $\nabla f(\mathbf{x}) = 0$. If the Hessian H of f has no zero eigenvalue, then the stationary point is *hyperbolic* and the behavior of the flow is locally determined by H: the distance from the stationary point changes like $\exp(-\mu t)$, where μ is the smallest eigenvalue of H. Away from stationary points, the flow moves like $t\|\nabla f(\mathbf{x})\|$, and the Hessian plays no role.

2.1. Measuring stability transverse to the flow on an invariant set.

To determine the stability of the gradient flow transverse to an invariant set, we use the following procedure:

1. Compute the Hessian at $\mathbf{x}$, denoted by $H(\mathbf{x})$.
2. Apply a change of coordinates so that an orthogonal basis for the tangent space of the invariant set at $\mathbf{x}$ is in some specific coordinate directions.
3. Delete those coordinates to obtain the "transverse" Hessian, denoted by $H^{\perp}(\mathbf{x})$.
4. Compute the smallest eigenvalue of $H^{\perp}(\mathbf{x})$, and denote it as $\mu(\mathbf{x})$. If $\mu(\mathbf{x}) > 0$, then the flow is stable; if $\mu(\mathbf{x}) < 0$, then the flow is unstable; and if $\mu(\mathbf{x}) = 0$, then the flow is (linearly) neutral.

The invariant set could be the flow path itself, for which $\nabla f(\mathbf{x})$ is a basis for the tangent space, or another set, such as some symmetry set.

To compare the stability of the flow at two different points, we will use the geometric measure

$$h(\mathbf{x}) = \frac{\mu(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \quad \in [-\infty, \infty], \tag{2.2}$$

which measures (in)stability with respect to distance along the flow, whereas $\mu(\mathbf{x})$ measures (in)stability with respect to time. For visualization purposes, we use

$$\tilde{h}(\mathbf{x}) = \frac{1}{2}\left(1 - \frac{h(\mathbf{x})}{\sqrt{1 + (h(\mathbf{x}))^2}}\right) \quad \in [0, 1]\,, \tag{2.3}$$

and so $\tilde{h}(\mathbf{x}) \in [0, 1/2)$ indicate stability and $\tilde{h}(\mathbf{x}) \in (1/2, 1]$ indicate instability.

2.2. Algorithm progress rate.

Recall that the gradient flow near a stationary point causes the distance from the stationary point to change like $\exp(-\mu t)$, where μ is the smallest eigenvalue of the Hessian. For an algorithm, such as the gradient-descent minimization algorithm

with line search, the rate of progress near a minimum or saddle also depends on η, the maximum eigenvalue of H. In both cases it is the ratio μ/η, rather than μ itself, that matters, with values near 0 indicating slow progress. For a saddle, if $\mu/\eta < -1$, then the algorithm escapes in one or two iterations so the saddle behaves more like a maximum.

Away from stationary points, the gradient flow moves like $t\|\nabla f(\mathbf{x})\|$. Under the assumption that the minimum eigenvalue of the transverse Hessian (with respect to the flow itself) from subsection 2.1 is positive, algorithm progress is governed by the ratio $2\|\nabla f(\mathbf{x})\|/\eta$, where η is the maximum eigenvalue of the transverse Hessian.

2.3. Converting rates to estimated times. In order to compare two points $\mathbf{x}_1$ and $\mathbf{x}_2$, it is advantageous to have a quantity that accounts for both the size of the objective function and size of its gradient. The quantity

$$s(\mathbf{x}) = \frac{f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|^2} \tag{2.4}$$

incorporates this information and can be interpreted as an estimate on the time it would take the gradient flow from $\mathbf{x}$ to reach a root (which is also a minimum) of f. For an algorithm, we replace the flow speed $\|\nabla f(\mathbf{x})\|$ by the algorithm progress rate $2\|\nabla f(\mathbf{x})\|/\eta$, to obtain

$$v(\mathbf{x}) = \frac{\eta}{2} s(\mathbf{x}) = \frac{f(\mathbf{x})\eta}{2\|\nabla f(\mathbf{x})\|^2} \quad \text{if } 0 < \mu \tag{2.5}$$

as an estimate of the algorithm time, with no estimate when $\mu < 0$. For visualization purposes, we use

$$\tilde{s}(\mathbf{x}) = \frac{s(\mathbf{x})}{1 + s(\mathbf{x})} \quad \in [0, 1], \tag{2.6}$$

$$\tilde{v}(\mathbf{x}) = \frac{v(\mathbf{x})}{1 + v(\mathbf{x})} \quad \in [0, 1]\,. \tag{2.7}$$

3. A model problem: Fitting a rank-2 tensor with a rank-2 tensor. In this section we sketch the analysis of fitting $T = (1.2)$ by $G_2 = (1.4)$ while deferring most formulas and proofs to Appendix A. For this model problem, the regularized least-squares error function $f(\mathbf{x}) = \|T - G\|^2 + \lambda \sum_{l=1}^{2} \| \bigotimes_{i=1}^{d} G_i^l \|^2$ becomes

$$E_\lambda(a, \boldsymbol{\alpha}, b, \boldsymbol{\beta}) = \|T - G_2\|^2 + \lambda \left(a^2 + b^2\right)\,, \tag{3.1}$$

with $\lambda \geq 0$.

In subsection 3.1 we show how to remove the scalars (a, b) from the analysis by making them fast variables. In subsection 3.2 we describe the behavior near the diagonal $\boldsymbol{\beta} = \boldsymbol{\alpha}$. In subsection 3.3 we consider the case when both T and G_2 are symmetric in direction, meaning that $\boldsymbol{\phi} = \phi\mathbf{1}$, $\boldsymbol{\alpha} = \alpha\mathbf{1}$, and $\boldsymbol{\beta} = \beta\mathbf{1}$. In subsection 3.4 we consider the case when T is symmetric in direction ($\boldsymbol{\phi} = \phi\mathbf{1}$) and G_2 satisfies one of two partial symmetries.

3.1. Expressing the error, gradient, and Hessian as a function of the angles by using fast variables. The dependence of $E_\lambda(a, \boldsymbol{\alpha}, b, \boldsymbol{\beta})$ in (3.1) on (a, b) is a distraction. Given $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, the value of (a, b) that minimizes $E_\lambda(a, \boldsymbol{\alpha}, b, \boldsymbol{\beta})$ is the least-squares solution to the system

$$\begin{bmatrix} \bigotimes_{i=1}^d \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix} & \bigotimes_{i=1}^d \begin{bmatrix} \cos(\beta_i) \\ \sin(\beta_i) \end{bmatrix} \\ \sqrt{\lambda} \bigotimes_{i=1}^d \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix} & 0 \\ 0 & \sqrt{\lambda} \bigotimes_{i=1}^d \begin{bmatrix} \cos(\beta_i) \\ \sin(\beta_i) \end{bmatrix} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix}. \tag{3.2}$$

Applying the transpose of the matrix on the left yields the normal equations

$$\begin{bmatrix} 1+\lambda & \prod_{i=1}^d \cos(\alpha_i - \beta_i) \\ \prod_{i=1}^d \cos(\alpha_i - \beta_i) & 1+\lambda \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} n(\boldsymbol{\alpha}) \\ n(\boldsymbol{\beta}) \end{bmatrix}, \quad \text{where} \tag{3.3}$$

$$n(\boldsymbol{\alpha}) = \left\langle \bigotimes_{i=1}^d \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix}, T \right\rangle = \text{(A.5)}. \tag{3.4}$$

Since the matrix is 2×2, we can explicitly invert it to obtain formulas for (a, b); the inverse exists except when $\lambda = 0$ and $\boldsymbol{\alpha} = \boldsymbol{\beta}$. By assuming (a, b) always take these optimal values, we can write the error as $E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Since $E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is known explicitly, we can then write down formulas for the gradient $\nabla E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$, which has two blocks corresponding to $\frac{\partial}{\partial \alpha_j} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $\frac{\partial}{\partial \beta_j} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Similarly, we can write down formulas for the Hessian, which has blocks corresponding to $\frac{\partial^2}{\partial \alpha_j \partial \alpha_k} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$, $\frac{\partial^2}{\partial \alpha_j \partial \beta_k} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$, $\frac{\partial^2}{\partial \beta_j \partial \alpha_k} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$, and $\frac{\partial^2}{\partial \beta_j \partial \beta_k} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$. All these formulas are given in Appendices A.2 and A.3. When fitting with G_1, the same process was used in [17] to reduce the error from $E_\lambda(a, \boldsymbol{\alpha})$ to $E_\lambda(\boldsymbol{\alpha})$.

Remark 3.1. The use of fast variables is a tool for understanding, *not* a proposed algorithm.

Remark 3.2. If one expresses G_2 in Euclidean rather than angular coordinates, runs the standard ALS algorithm, and then expresses the result back in the angular form (1.4), then the resulting (a, b) will have their optimal values as described above. This is a side effect of the action of ALS as an alternating orthogonal projection; see [29, section 4.1].

3.2. Behavior near the diagonal $\boldsymbol{\beta} = \boldsymbol{\alpha}$. If $\lambda > 0$, then we can simply set $\boldsymbol{\beta} = \boldsymbol{\alpha}$ and find that the two terms in G_2 are identical. Each is half of the G_1 obtained using $E_{\lambda/2}(\boldsymbol{\alpha})$.

If $\lambda = 0$ and $\boldsymbol{\beta} = \boldsymbol{\alpha}$, then the determinant of the matrix in (3.3) is zero so $E_0(\boldsymbol{\alpha}, \boldsymbol{\alpha})$ is undefined. Fixing a vector $\mathbf{v}$ with $\|\mathbf{v}\| = 1$, one can show that

$$\lim_{\epsilon \to 0} E_0(\boldsymbol{\alpha}, \boldsymbol{\alpha} + \epsilon \mathbf{v}) = 1 - \left(n^2(\boldsymbol{\alpha}) + \left(\sum_{i=1}^d v_i n_i(\boldsymbol{\alpha}) \right)^2 \right), \tag{3.5}$$

where $n(\boldsymbol{\alpha}) =$ (3.4) = (A.5) and $n_i(\boldsymbol{\alpha}) =$ (A.7) is its partial derivative with respect to α_i. The limit exists and is finite for any $\mathbf{v}$. If $\nabla n(\boldsymbol{\alpha}) \neq \mathbf{0}$, then the value of the limit depends on $\mathbf{v}$. Thus $E_0(\boldsymbol{\alpha}, \boldsymbol{\beta})$, considered as a function of $\boldsymbol{\beta}$ for fixed $\boldsymbol{\alpha}$, has an essential discontinuity at $\boldsymbol{\beta} = \boldsymbol{\alpha}$. The minimum of the limit (3.5) with respect to $\mathbf{v}$ can be computed as

$$1 - \left(n^2(\boldsymbol{\alpha}) + \sum_{i=1}^{d} n_i^2(\boldsymbol{\alpha}) \right). \tag{3.6}$$

If $\langle \mathbf{v}, \nabla n(\boldsymbol{\alpha}) \rangle \neq 0$, then $\lim_{\epsilon \to 0}(|a|, |b|) = (\infty, \infty)$.

These statements are formulated and proven in Lemmas A.1 to A.3 in Appendix A.4.

3.3. The case of symmetric T and G_2. If $\boldsymbol{\phi} = \phi\mathbf{1}$, $\boldsymbol{\alpha} = \alpha\mathbf{1}$, and $\boldsymbol{\beta} = \beta\mathbf{1}$, then T and G_2 are symmetric under permutations of directions and this property is preserved as G_2 changes under the gradient flow. We can then analyze a problem in only six variables $(d, z, \phi, \lambda, \alpha, \beta)$. Selecting (d, z, ϕ) to specify T and λ to specify the error function reduces the problem to two variables (α, β).

The first gain from the reduction to two variables is that analysis becomes more tractable. In Appendix A.5 we given the analysis in detail. For the moment we only highlight one result that we will refer to in section 4. When $\lambda = 0$ and $z = \pm 1$, the analysis allows us to compute the eigenvalues of the Hessian at the solutions $(\alpha, \beta) = (0, \phi)$ and $(\alpha, \beta) = (\phi, 0)$. We then show that as $\phi \to 0^+$ the condition number of the Hessian goes to infinity and the algorithm progress rate of subsection 2.2 goes to zero. Thus for small enough ϕ there will be a terminal swamp and it will worsen as $\phi \to 0^+$.

The second gain from this reduction is that we can plot $E_\lambda(\alpha, \beta)$ and other quantities and *see* what is going on. A key to the formatting of these plots is given in Figure 3.1. We have the following quantities to plot:

1. The error $E_\lambda(\alpha, \beta)$ as a function of (α, β).
2. The transverse stability indicator from subsection 2.1 in the form $\tilde{h}(\alpha, \beta) =$ (2.3). Values below 1/2 indicate stability, and values above 1/2 indicate instability; the contour at 1/2 is highlighted in black. When stability is indicated, a perturbation of the flow remains near the plotted set so it is meaningful to follow the gradient flow path in the plot of $E_\lambda(\alpha, \beta)$.
3. The estimated flow time, as determined in subsection 2.3, in the form $\tilde{s}(\alpha, \beta) =$ (2.6). Large values indicate swamp candidates.
4. The estimated algorithm time, as determined in subsection 2.3, in the form $\tilde{v}(\alpha, \beta) =$ (2.7). Large values indicate swamp candidates.
5. The mapped amplitude $\tilde{a}(\alpha, \beta) = (a^2 + b^2)/(1 + a^2 + b^2)$. Small values indicate that G_2 itself is small, and large values indicate cancellation within G_2.
6. The limiting error on the diagonal $\lim_{\beta \to \alpha} E_\lambda(\alpha, \beta)$ as a function of α. For $\lambda = 0$, we also plot $E_0(\alpha)$ for G_1 (dashed), thus illustrating the gap between the two; see the discussion in subsection 3.2. The horizontal axis is $\alpha \in [\phi/2 - \pi/2, \phi/2 + \pi/2]$, with 0 and ϕ marked by vertical lines, and the vertical axis is $[0, 1]$.

In the supplementary material (M118138_01.pdf [local/web 33.5MB]) we present a gallery of plots of these six quantities using a wide selection of parameter choices. We now present and interpret these plots for a single choice of (d, z, ϕ, λ) with the goal of training the viewer in

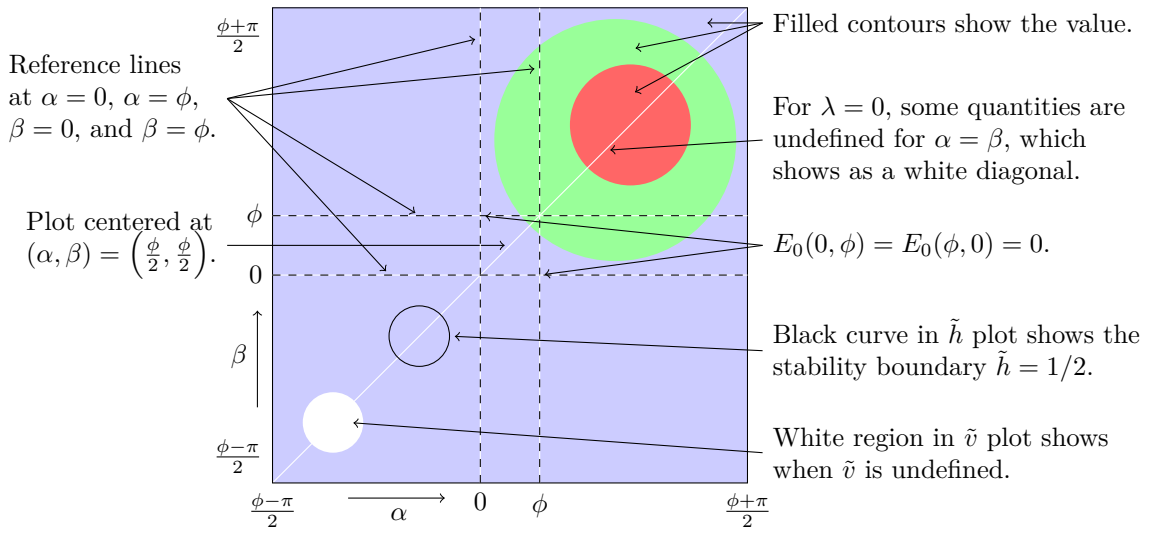

Figure 3.1. *Plotting format for the error E_λ, estimated flow time $\tilde{s}$, estimated algorithm time $\tilde{v}$, stability $\tilde{h}$, and amplitude measure $\tilde{a}$ as functions of (α, β) for a fixed value of (d, z, ϕ, λ). The plot is symmetric $\alpha \leftrightarrow \beta$ and periodic with period π in both α and β.*

how to interpret these plots. We select $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$ because we can reuse it in the example swamps in subsections 4.1 and 4.2.

In Figure 3.2 we show these six plots for $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$. We observe the following:

- There is a gap between $\lim_{\beta \to \alpha} E_\lambda(\alpha, \beta)$ and $E_0(\alpha)$ except when $\alpha \in \{\phi/2, \phi/2 \pm \pi/2\}$.
- The error E_0 has a single maximum at $(\phi/2 \pm \pi/2, \phi/2 \pm \pi/2)$ and two minima at the solutions $\{(0, \phi), (\phi, 0)\}$. There is a saddle at $(\phi/2, \phi/2)$. As long as it stays in this symmetry plane, the flow will move rapidly off the maximum into the valleys along $\alpha \approx \phi/2$ or $\beta \approx \phi/2$. It will then move along the valley to $(\alpha, \beta) = (0, \phi)$ or $(\alpha, \beta) = (\phi, 0)$.
- The stability indicator $\tilde{h}$ shows transverse stability along the valleys leading to the solutions. It shows transverse instability in some regions, such as $\alpha \approx \beta \approx \phi/4$, and so the flow passing through those regions would leave the plotting plane.
- To interpret the amplitude $\tilde{a}$ we first see that its value is around 0.4 at the solutions $\{(0, \phi), (\phi, 0)\}$, so that is its natural size. When it is small, such as at $(3\phi, -2\phi)$, neither term in G_2 is effective in helping approximate T, and so both are small. When it is large, such as the region when $\alpha \approx \beta$ and $\lim_{\beta \to \alpha} E_\lambda(\alpha, \beta) < E_0(\alpha)$, a better (but perhaps still not good) approximation can be obtained using large cancellation. Near the maximum, $\tilde{a}$ has small values when the approach is near the antidiagonal $\beta = \phi - \alpha$ but large values when the approach is near the diagonal $\alpha = \beta$.
- The flow time $\tilde{s}$ is large near the maximum and saddle; at such locations a nonzero error is divided by a zero gradient, and so the fact that it is large does not give any information. It is also large along the valley floor, indicating slow progress and a candidate swamp. Analysis of the Hessian at the solutions $\{(0, \phi), (\phi, 0)\}$ shows that

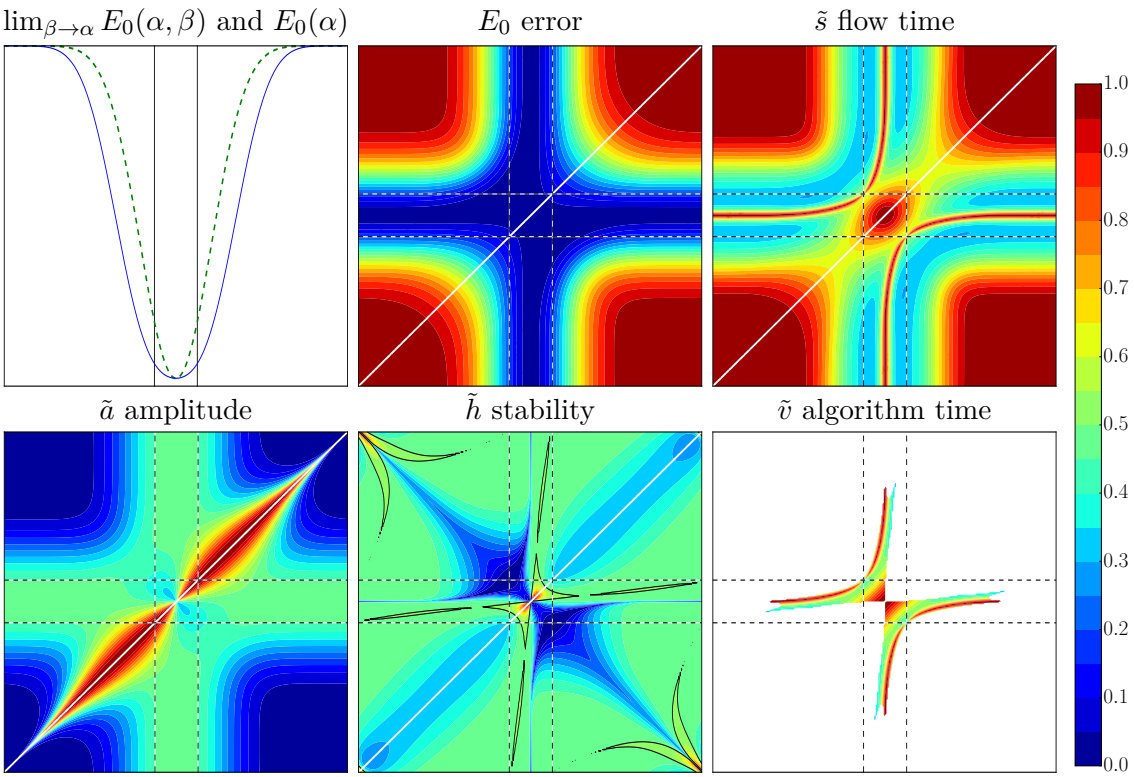

Figure 3.2. *Visualization of the case* $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$ *with symmetry* $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$. *The terminal swamp described in subsection 4.1 uses this case and maintains this symmetry. The transient swamp described in subsection 4.2 uses this case but moves off of this symmetry.*

the valley persists all the way to the solution and would worsen for smaller ϕ; see subsection 3.3 and Appendix A.5.3. Thus the slow progress is not transient and we have a candidate terminal swamp.

- The algorithm time $\tilde{v}$ is undefined in a large region, indicating that an algorithm starting there would immediately move elsewhere, possibly breaking symmetry. It is defined and large along the valley floor, again indicating a terminal swamp.

3.4. The case of symmetric T and partially symmetric G_2. We now consider the case where T is symmetric, with $\boldsymbol{\phi} = \phi\mathbf{1}$, but G_2 is only partially symmetric, with $\alpha_2 = \cdots = \alpha_d$ and $\beta_2 = \cdots = \beta_d$. The motivation for considering this partial symmetry is that it occurred in the numerical experiments in subsection 4.2. This partial symmetry is also consistent with the nonsymmetric saddle-point analysis in [17].

Selecting (d, z, ϕ) to specify T and λ to specify the error function reduces the problem to four variables $(\alpha_1, \alpha_d, \beta_1, \beta_d)$. We then separately consider two different further constraints, $\boldsymbol{\beta} \to \boldsymbol{\alpha}$ and $\boldsymbol{\beta} = \phi - \boldsymbol{\alpha}$, which also occurred in the numerical experiment in section 4. Both constraints reduce the problem to two variables. We again give plots for a selection of parameters in the supplementary material (M118138_01.pdf [local/web 33.5MB]) and plots for a single parameter choice here.

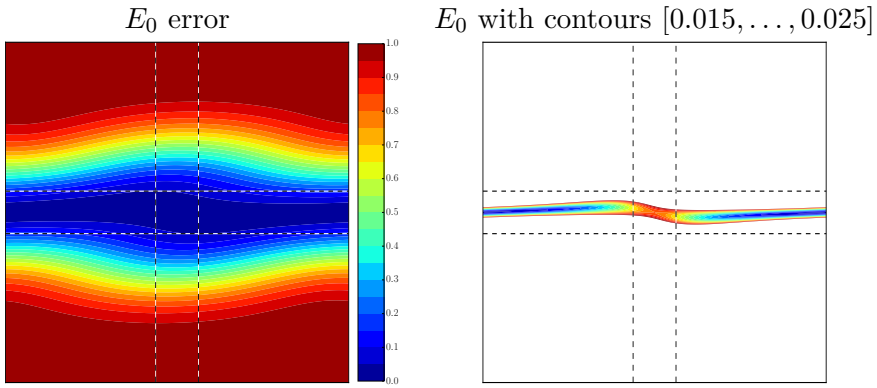

Figure 3.3. *The error $E_0(\alpha_1, \alpha_d)$ in the limit $\boldsymbol{\beta} \to \boldsymbol{\alpha}$ with $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$. The format is like Figure 3.1, except the horizontal axis is α_1 and the vertical axis is α_d. In the right-hand plot, the contours have been set at $[0.015, \ldots, 0.025]$ to better illustrate the location of the minimum; the plot is white when $E_0 > 0.025$. The transient swamp described in subsection 4.2 uses this case and passes through this symmetry.*

First, consider the symmetry with $\alpha_2 = \cdots = \alpha_d$ and $\boldsymbol{\beta} \to \boldsymbol{\alpha}$, with quantities expressed in terms of (α_1, α_d). For $\lambda = 0$, the limiting error $\lim_{\boldsymbol{\beta} \to \boldsymbol{\alpha}} E_0(\boldsymbol{\alpha}, \boldsymbol{\beta})$ taken from the optimal direction is given by (3.6) (and more explicitly in Lemma A.2). The quantities $\tilde{a}$, $\tilde{h}$, $\tilde{s}$, and $\tilde{v}$ are less meaningful and/or difficult to compute in this limit, and so we do not consider them. Figure 3.3 gives $E_0(\alpha_1, \alpha_d)$ for $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$. We observe that the minimum is at $(\alpha_1, \alpha_d) = (\phi/2 \pm \pi/2, \phi/2)$. If the flow stays near this plotting plane, then it will exit at this specific point. Since we do not have a transverse stability indicator in the limiting symmetry, we are not guaranteed to stay near this plotting plane. Note that the quantity $\lim_{\beta \to \alpha} E_0(\alpha, \beta)$ plotted in subsection 3.3 under the symmetry $\boldsymbol{\alpha} = \alpha \mathbf{1}$ corresponds to setting $\alpha_1 = \alpha_d$ in the current symmetry and thus appears on the diagonal in the current plot.

Second, consider the symmetry with $\alpha_2 = \cdots = \alpha_d$ and $\boldsymbol{\beta} = \phi - \boldsymbol{\alpha}$, with quantities expressed in terms of (α_1, α_d). If $z = \pm 1$, then $E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is symmetric under the map $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \mapsto (\phi - \boldsymbol{\beta}, \phi - \boldsymbol{\alpha})$, and thus the set with $\boldsymbol{\beta} = \phi - \boldsymbol{\alpha}$ is invariant under the gradient flow. We can then plot E_λ as well as the quantities $\tilde{a}$, $\tilde{h}$, $\tilde{s}$, and $\tilde{v}$. Figure 3.4 gives an array of plots for $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$. We observe the following:

- The amplitude $\tilde{a}$ is small in the region where the error is large. It is large from some directions near $(\alpha_1, \alpha_d) = (\phi/2 \pm \pi/2, \phi/2)$. Since the error and other quantities are π-periodic, this point corresponds to the diagonal $\boldsymbol{\alpha} = \boldsymbol{\beta}$, where one can have $(|a|, |b|) \to (\infty, \infty)$, as discussed in subsection 3.2.
- There are steep-sided valleys descending from $(\alpha_1, \alpha_d) = (\phi/2 \pm \pi/2, \phi/2)$ to the solutions at $(\alpha_1, \alpha_d) = (0, 0)$ and $(\alpha_1, \alpha_d) = (\phi, \phi)$. In these valleys, $\tilde{h}$ shows stability, $\tilde{s}$ shows large flow time, and $\tilde{v}$ shows large algorithm time. Higher-resolution versions of these plots show that the valleys exit $(\alpha_1, \alpha_d) = (\phi/2 \pm \pi/2, \phi/2)$ at a slant and the regions of stability, large flow time, and large algorithm time extend up to this point, which is an essential discontinuity.

Note that the diagonal $\alpha_1 = \alpha_d$ in the current symmetry corresponds to the antidiagonal $\beta = \phi - \alpha$ in the symmetry $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha \mathbf{1}, \beta \mathbf{1})$ of subsection 3.3 and the points $(\alpha_1, \alpha_d) \in$

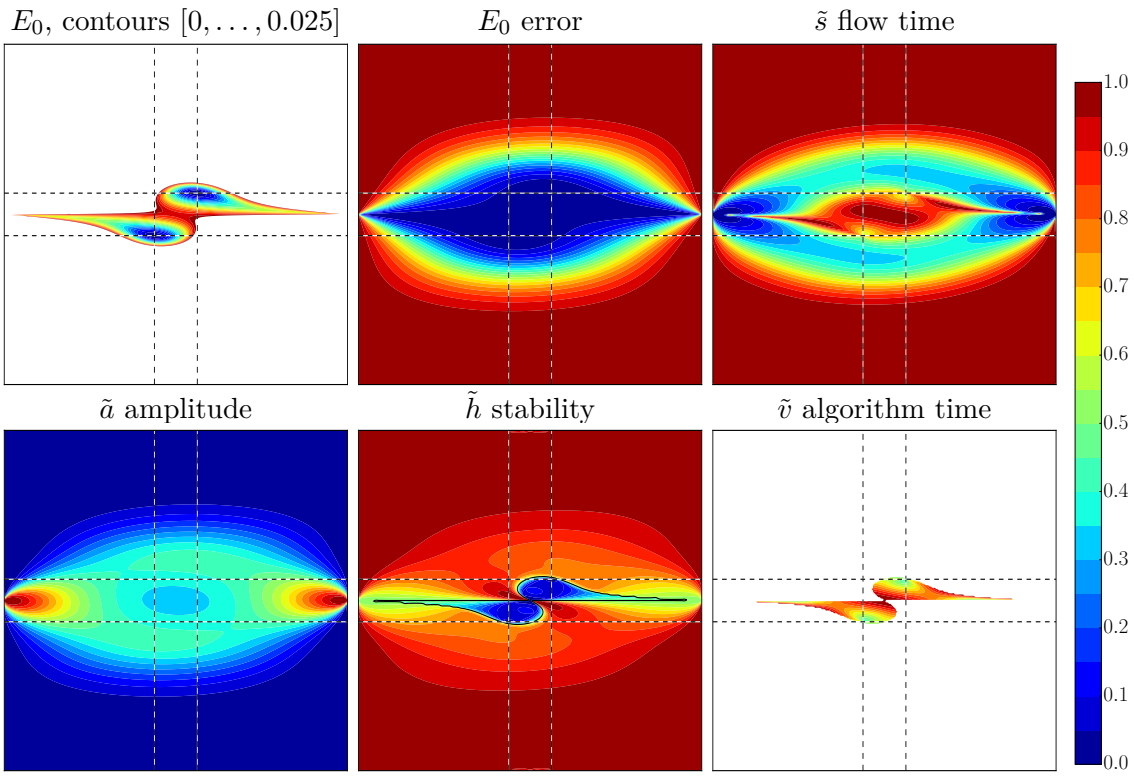

Figure 3.4. *Visualization of the case* $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$ *under the constraints* $\alpha_2 = \cdots = \alpha_d$ *and* $\boldsymbol{\beta} = \phi - \boldsymbol{\alpha}$ *with horizontal axis* α_1 *and vertical axis* α_d*. The transient swamp described in subsection 4.2 uses this case and passes through this symmetry.*

$\{(0,0), (\phi,\phi)\}$ correspond to $(\alpha, \beta) \in \{(0,\phi), (\phi,0)\}$.

4. Example swamps. In this section we analyze the dynamics of several swamps. We run the standard version of ALS (see, e.g., [29]) and then express the results in terms of $(a, \boldsymbol{\alpha}, b, \boldsymbol{\beta})$. As noted in Remark 3.2, ALS automatically sets (a, b) to their optimal values.

4.1. A terminal swamp. In this section we describe in detail the terminal swamp shown in Figure 1.1, which was produced using the parameters $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$. This swamp was discovered by looking at plots as described in subsection 3.3, which use the symmetry $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$, and noticing the potential terminal swamp in Figure 3.2.

This terminal swamp was produced using starting point $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = ((3/5)\phi\mathbf{1}, ((2/5)\phi + \pi/2)\mathbf{1})$. The swamp can be made more severe by reducing ϕ; for example, setting $\phi = \pi/16$ increases the number of iterations from 270 to 3650. The flow path of this terminal swamp consists of two phases, which we illustrate as columns in Figure 4.1. Both columns use constraint $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$, axes $(x, y) = (\alpha, \beta)$, and E_0 from Figure 3.2:

1. The flow descends rapidly to $\alpha_i \approx \phi/2$ and $\beta_i \approx 2\phi$ and recovers $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$ symmetry.
2. The flow moves slowly along a curve on the valley floor until $(\alpha, \beta) = (0, \phi)$. Progress is slow because the condition number of the Hessian as analyzed in Appendix A.5.3 is

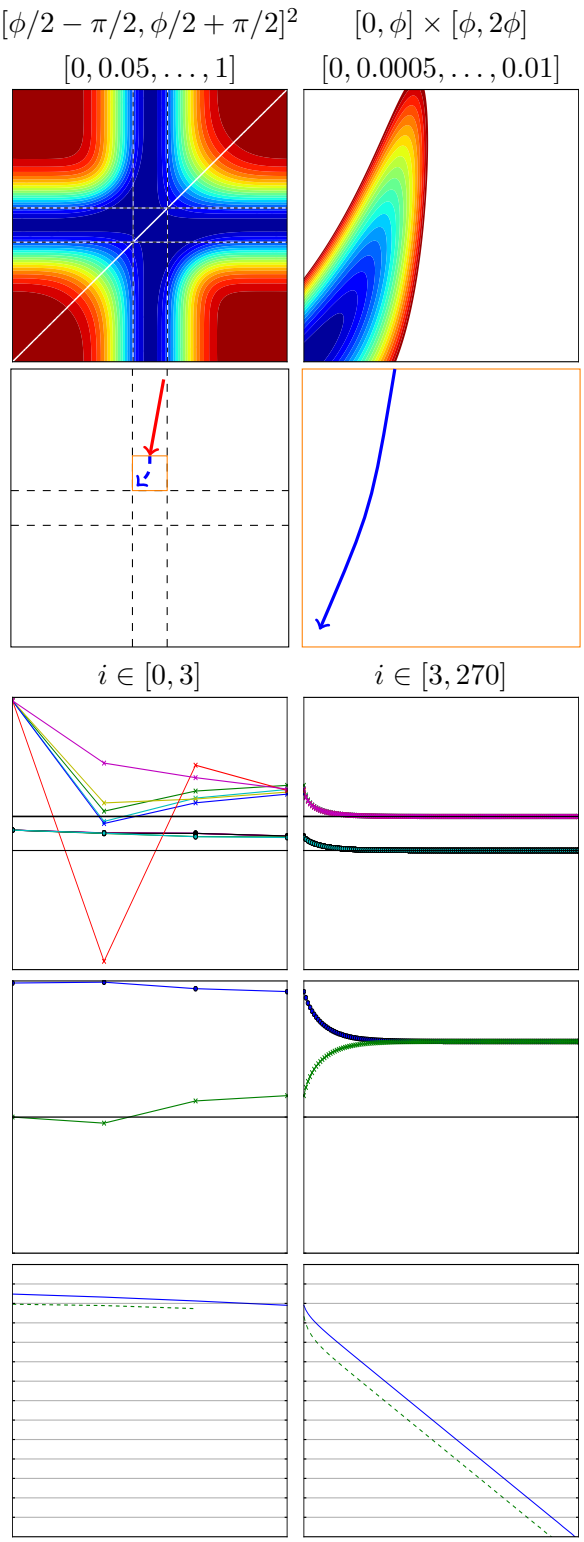

Figure 4.1. *Illustration of a terminal swamp flow path for* $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$.

large; for these parameters it is $\eta^{\perp}/\mu^{\parallel} \approx 48.5$.

4.2. A transient swamp. In this section we describe the dynamics of the transient swamp shown in Figure 1.2. It uses the same parameters $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$ as the terminal swamp in subsection 4.1 but a different initial point. The path passes through all three of the symmetries described in subsections 3.3 and 3.4 and illustrated in Figures 3.2 to 3.4.

This transient swamp was produced using starting point $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = ((\phi/2 + \pi/2)\mathbf{1}, (\phi/2 + \pi/2 - 1/5)\mathbf{1})$. The severity of the swamp can be changed by varying the starting point or parameters; for example, setting $\boldsymbol{\beta} = (\phi/2 + \pi/2 - 1/10)\mathbf{1}$ produces a transient swamp needing about 2200 iterations, setting $\phi = \pi/16$ produces one needing about 16000 iterations, and setting $\phi = \pi/3$ shortens the swamp to 37 iterations.

The flow path of the transient swamp in Figure 1.2 consists of four phases, which we illustrate as four columns in Figure 4.2:

1. The flow starts in the symmetric set $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$ studied in subsection 3.3 and visualized in Figure 3.2. Starting with $\boldsymbol{\alpha} \approx \boldsymbol{\beta} \approx (\phi/2 + \pi/2)\mathbf{1}$, the flow proceeds toward $\boldsymbol{\alpha} \approx \boldsymbol{\beta} \approx (\phi/2)\mathbf{1}$. Between $\phi\mathbf{1}$ and $(\phi/2)\mathbf{1}$ it encounters transverse instability with respect to $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$, as indicated in the $\tilde{h}$ plot in Figure 3.2. In this phase the flow and algorithm progress are fast. The error reduces significantly, and it is not apparent from the error that the flow is entering a swamp.
2. Since $\boldsymbol{\alpha} \approx \boldsymbol{\beta}$, the flow encounters an error landscape similar to the $\boldsymbol{\beta} \to \boldsymbol{\alpha}$ limit plotted in Figure 3.3. In all but one direction, the angles flow until $\alpha_i \approx \beta_i \approx \phi/2$. In the remaining direction, which we will label $i = 1$, the angles flow to $\alpha_1 \approx \beta_1 \approx \phi/2 \pm \pi/2$. In this phase the flow and algorithm progress are also fast.
3. The angles $\alpha_i \approx \beta_i \approx \phi/2$ satisfy $\beta_i \approx \phi - \alpha_i$; due to the π-periodicity of the error, $\alpha_1 \approx \beta_1 \approx \phi/2 \pm \pi/2$ also satisfies $\beta_1 \approx \phi - \alpha_1$. The flow then encounters the landscape illustrated in Figure 3.4; we zoom in to show $(\alpha_1, \alpha_d) \in [\phi/2 - \pi/2, 3\phi/2 - \pi/2] \times [0, \phi]$. Maintaining $\boldsymbol{\beta} \approx \phi - \boldsymbol{\alpha}$, and $\alpha_2 \approx \cdots \approx \alpha_d$, α_1 increases to $\approx 3\phi/2 - \pi/2$ and $\alpha_2 \approx \cdots \approx \alpha_d$ decrease slightly. The flow and algorithm progress are very slow for many iterations. The change in error stays very small for many iterations but is increasing at an increasing rate. This phase is the transient swamp itself and is the descent from an essential saddle. Figure 1.4 shows a centered version of this essential saddle, with $\alpha_1 \in [\phi/2 - 3\pi/4, \phi/2 - \pi/4]$ and $\alpha_d \in [\phi/2 - \pi/4, \phi/2 + \pi/4]$.
4. All α_i flow rapidly to 0, which makes $\beta_i = \phi$ and gives the solution. Although this is the same minimum that caused the terminal swamp in subsection 4.1, the approach is orthogonal to the eigenvectors corresponding to $\mu^{\parallel}$ and $\eta^{\perp}$, and so the relevant condition number from Appendix A.5.3 is $\eta^{\parallel}/\mu^{\perp} \approx 1.6$.

4.3. A double swamp. In this section we describe a swamp that has a transient part followed by a terminal part. The terminal part is caused by an ill-conditioned Hessian, as was the terminal swamp in subsection 4.1. The transient part is similar to that in subsection 4.2 in that the flow is descending along a valley away from an essential saddle. This swamp was produced using $(d, z, \phi, \lambda) = (6, -1, \pi/8, 0)$.

The path maintains the approximate symmetry $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$, and so in Figure 4.3 we plot using the current parameters and that symmetry. We observe the following:

- The solutions $\{(0, \phi), (\phi, 0)\}$ are contained within a valley along the antidiagonal.

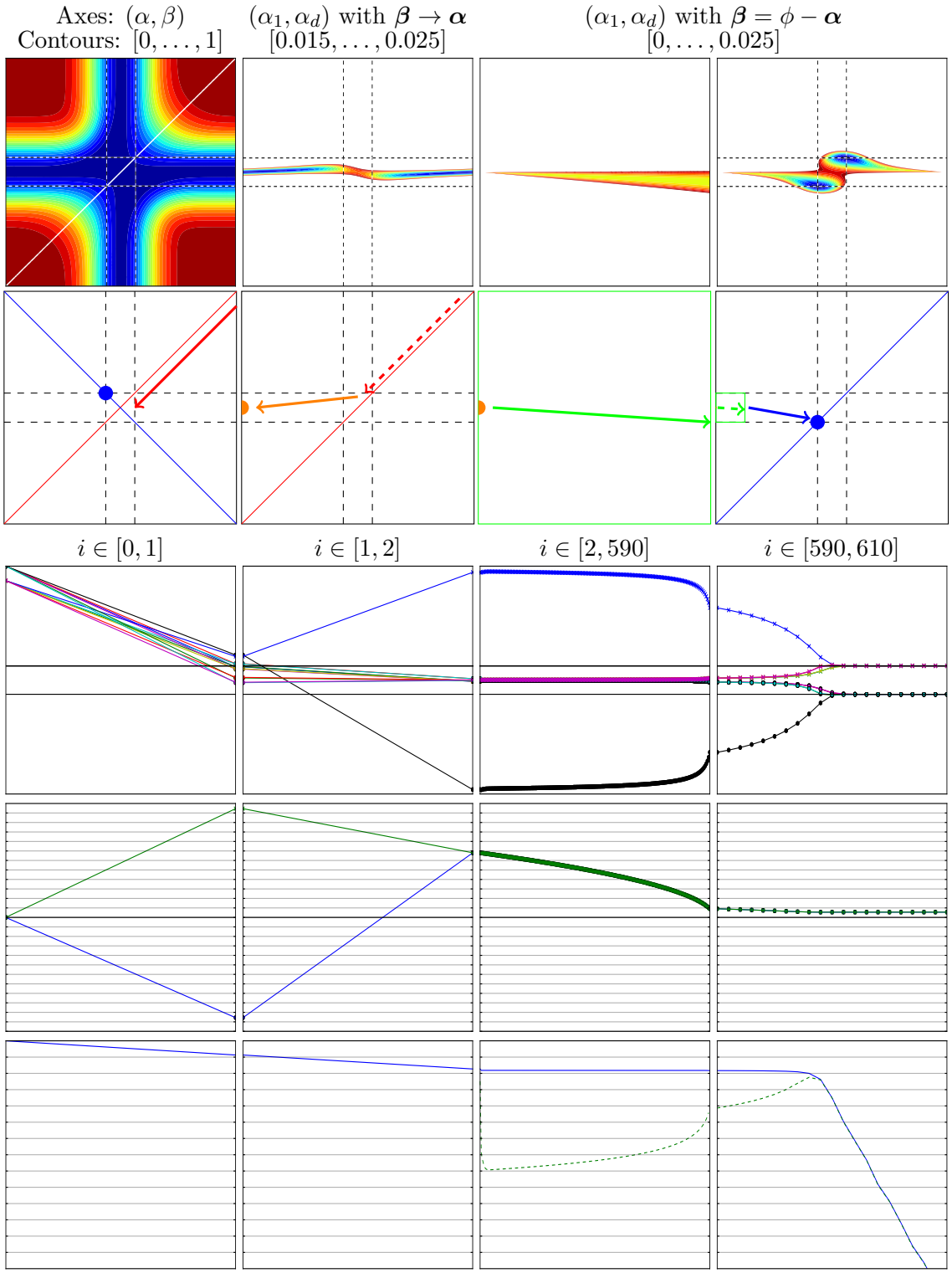

Figure 4.2. *A transient swamp flow path for* $(d, z, \phi, \lambda) = (6, 1, \pi/8, 0)$*, formatted like Figure 4.1.*

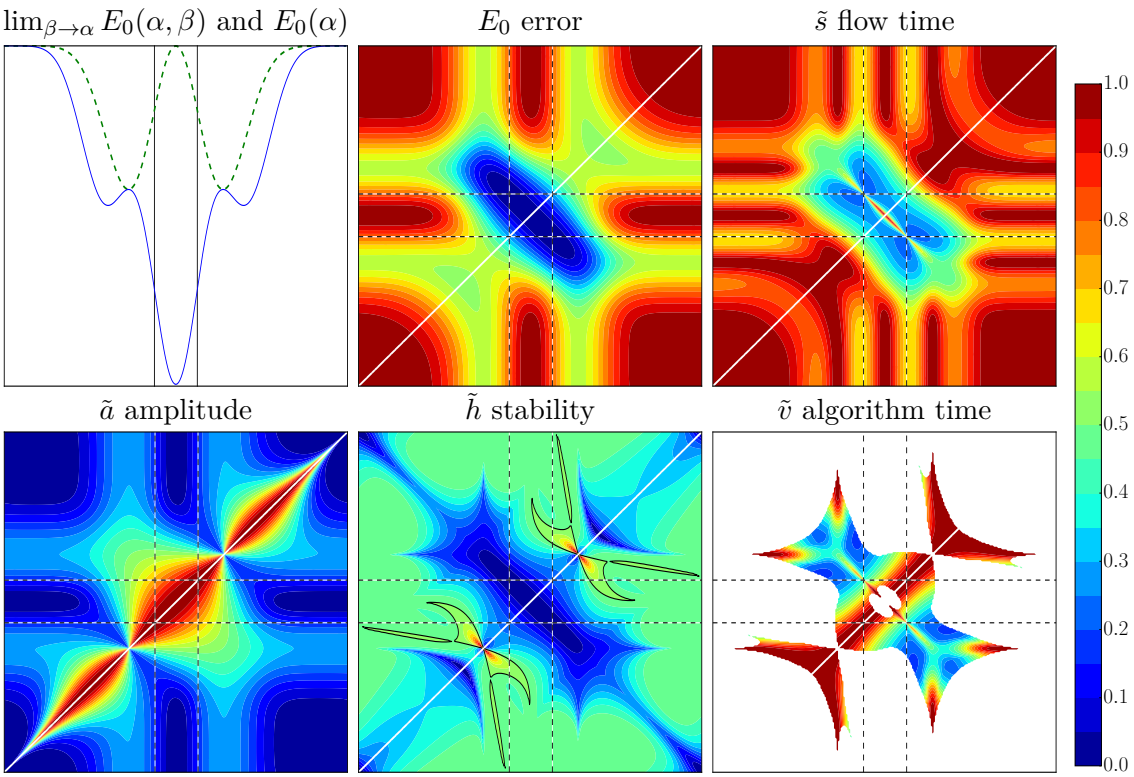

Figure 4.3. *Visualization of the case* $(d, z, \phi, \lambda) = (6, -1, \pi/8, 0)$ *with symmetry* $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$*, on which occurs the double swamp in Figure 4.5 and the local infimum in subsection 4.5.1.*

There is transverse stability in that region, and so the flow would move rapidly into the valley and then slowly along it.

- There is a saddle-like feature in the diagonal limit $\beta \to \alpha = \phi/2$ and a stripe with large flow time $\tilde{s}$ leading from it to the solutions. There is a less-severe stripe with large flow time on the opposite sides of the solutions. The algorithm time shows similar stripes.

Part of the time, the path also maintains the symmetry $\alpha_2 = \cdots = \alpha_d$ and $\boldsymbol{\beta} = \phi - \boldsymbol{\alpha}$, and so in Figure 4.4 we plot with that symmetry. We observe steep-sided valleys descending from $(\alpha_1, \alpha_d) \approx (\phi/2, \phi/2)$ to the solutions at $(\alpha_1, \alpha_d) = (0, 0)$ and $(\alpha_1, \alpha_d) = (\phi, \phi)$ with corresponding large flow and algorithm times. Higher-resolution versions of these plots show that the valleys exit $(\alpha_1, \alpha_d) \approx (\phi/2, \phi/2)$ on the diagonal and the regions of stability, large flow time, and large algorithm time extend all the way to this point. These valleys better explain the large flow and algorithm times on the antidiagonal in Figure 4.3 than the E_0 error plot in Figure 4.3 does.

This swamp used starting point $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = ((3/4)\phi\mathbf{1}, \phi\mathbf{1})$. The swamp can be made more severe by making α closer to β or by reducing ϕ; for example, setting $\alpha = (9/10)\beta$ increases the number of iterations from 590 to 1960 and setting $\phi = \pi/16$ increases them from 590 to 9800. The flow path of this swamp consists of three phases, which we illustrate in Figure 4.5.

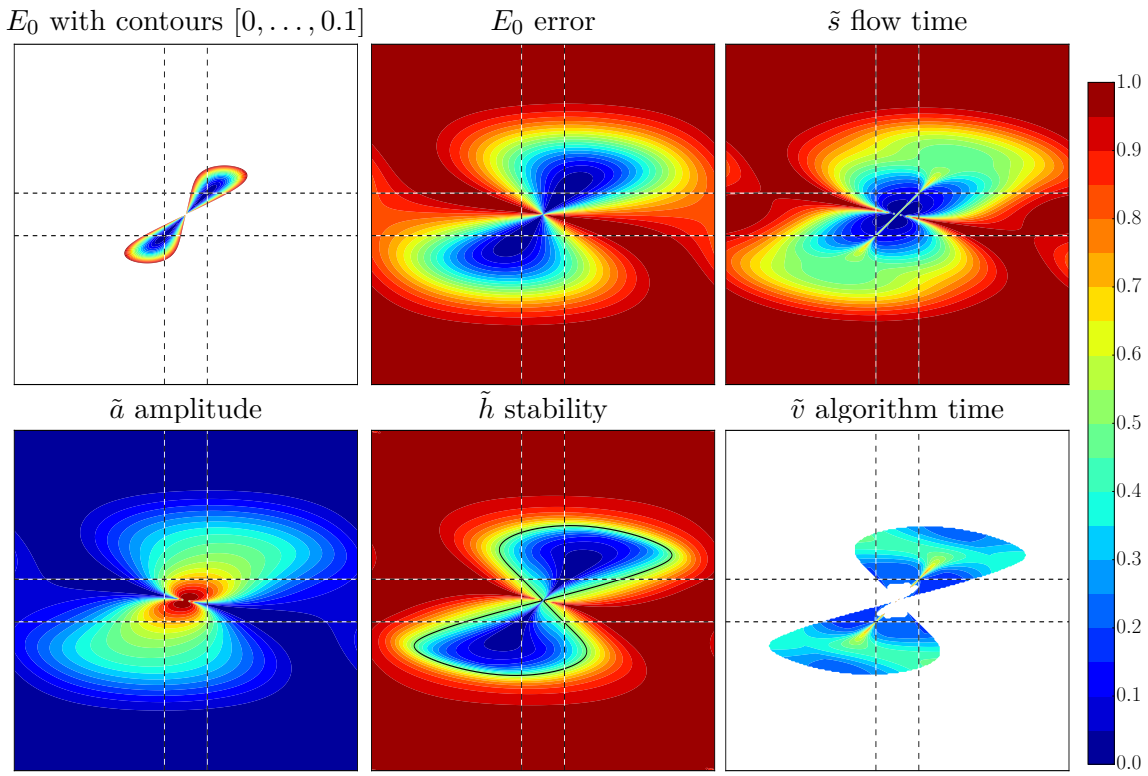

Figure 4.4. *Visualization of the case* $(d, z, \phi, \lambda) = (6, -1, \pi/8, 0)$ *under the constraints* $\alpha_2 = \cdots = \alpha_d$ *and* $\boldsymbol{\beta} = \boldsymbol{\phi} - \boldsymbol{\alpha}$ *with horizontal axis* α_1 *and vertical axis* α_d*. This symmetry shows the essential saddle causing the transient portion of the double swamp in Figure 4.5.*

In the second phase, we display both symmetries:

1. The flow descends rapidly to $\alpha_i \approx \phi/3$ and $\beta_i \approx 2\phi/3$, recovers $\boldsymbol{\alpha} = \alpha\mathbf{1}$ symmetry, and gains $\boldsymbol{\beta} = \boldsymbol{\phi} - \boldsymbol{\alpha}$ symmetry. The E_0 plot is from Figure 4.3, zoomed to $(\alpha, \beta) \in [0, \phi]^2$.
2. The flow moves along a valley along the antidiagonal $\beta = \phi - \alpha$, with α decreasing slowly. Change in error is small but increasing at an increasing rate. The essential saddle at $\alpha = \phi/2$ is the origin of this valley. The E_0 plots are from Figures 4.3 and 4.4, zoomed to $[0, \phi]^2$.
3. Once $\alpha \approx 0$, both the error and the change in error decrease at a slow but linear rate. Exactly as in the terminal swamp in subsection 4.1, the condition number of the Hessian is $\eta^{\perp}/\mu^{\parallel} \approx 48.5$. The E_0 plot is from Figure 4.3, zoomed to $(\alpha, \beta) \in [0, \phi/10] \times [9\phi/10, \phi]$.

4.4. A swamp from Paatero. In [34], Paatero analyzes aspects of the geometry of rank-2 tensors within the set of all $2 \times 2 \times 2$ tensors. In [34, section 6], he considers an example in which the target tensor and starting approximation are both of rank 2, but a region of rank-3 tensors blocks the direct path between them. The target tensor is

$$\left[\begin{array}{cc|cc} 0 & 1 & 30 & 0 \\ 1 & x & 0 & y \end{array}\right] = \left[\begin{array}{cc|cc} 0 & 1 & 30 & 0 \\ 1 & 0.25 & 0 & -0.28 \end{array}\right], \tag{4.1}$$

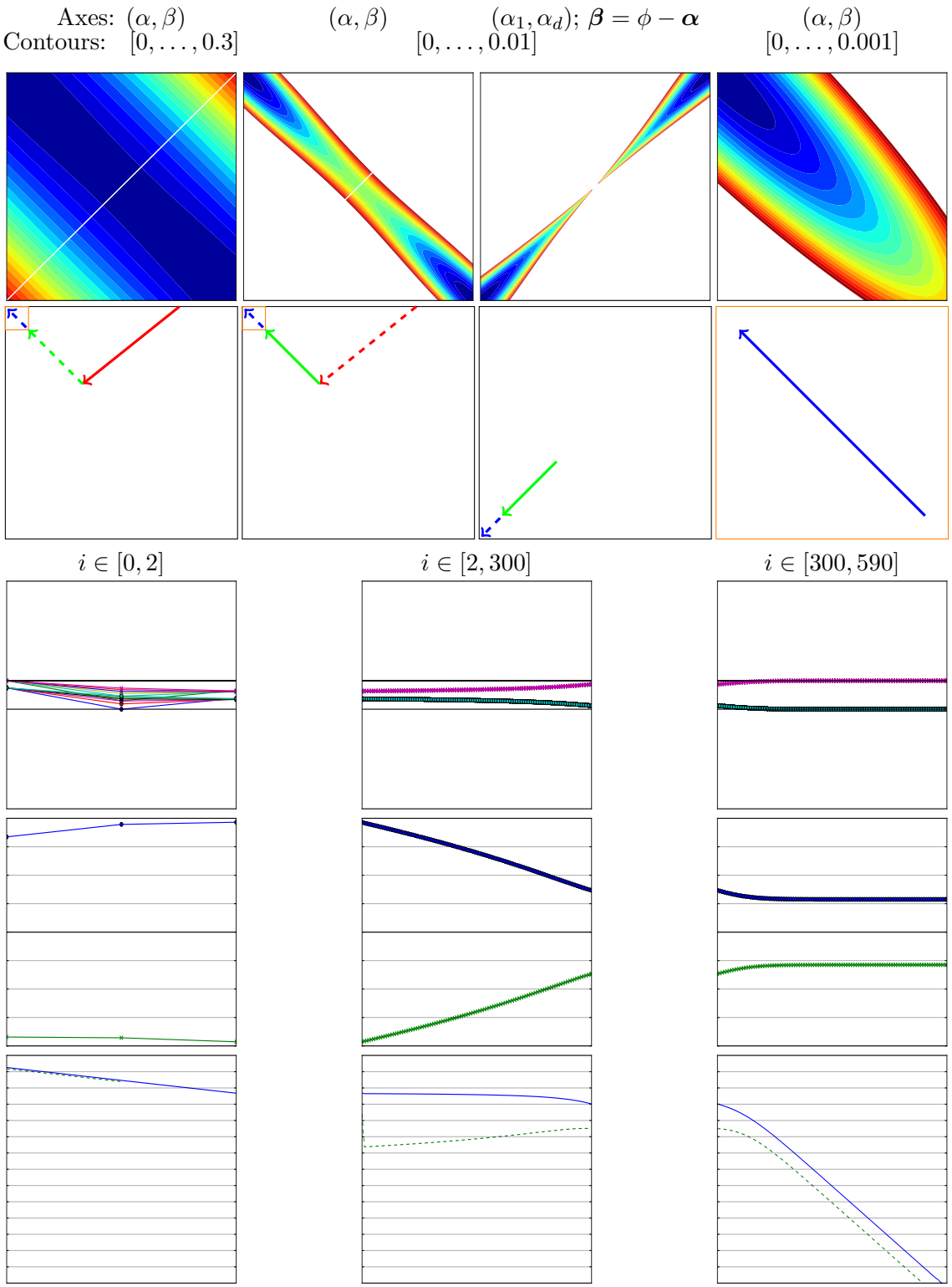

Figure 4.5. *A double swamp flow path for* $(d, z, \phi, \lambda) = (6, -1, \pi/8, 0)$*, formatted like Figure 4.1.*

and the starting tensor is of the same form but with $x = -0.23$ and $y = -0.047$. Using the rank-2 formula from [18, p. 477], we can express the target exactly with rank 2 and then express it in the form (1.2) with $z \approx -0.284$ and $\boldsymbol{\phi} \approx (0.156, 0.156, 0.469)$. Applying the same transformations to the starting point yields $\boldsymbol{\alpha}_0 \approx (-0.053, -0.053, -0.310)$ and $\boldsymbol{\beta}_0 \approx (-0.265, -0.265, -1.525)$. Applying ALS results in a swamp, which Paatero explains geometrically, in four stages, as follows:

1. There are 50 iterations of rapid advance toward the degenerate domain, which has rank 3 and thus acts as a barrier.
2. Then there are 150 iterations of slow advance toward the degenerate domain in an attempt to get around it. The approximation becomes more degenerate.
3. Next there are 190 iterations along the far side of the degenerate domain to a point along the shortest line segment from the target to the degenerate domain.
4. Finally, there are many iterations of slow convergence along this segment. Progress is slow because the approximation possesses degeneracy acquired in stage 2.

In this context, a *degenerate* tensor is a rank-3 tensor that is the limit of rank-2 tensors. A rank-2 tensor near a degenerate tensor will have large scalars and significant cancellation (as in Lemma A.3) and is (qualitatively) said to possess degeneracy.

In the first row of Figure 4.6 we illustrate these stages with diagrams corresponding to [34, Figure 1]. The horizontal axis is $x \in [-3.2, 3.2]$ and the vertical axis $y \in [-0.5, 0.14]$, with x and y from (4.1) and the other entries fixed as in (4.1). The starting and ending tensors satisfy (4.1) and thus are in this plotting plane, whereas intermediate tensors will be outside the plotting plane and we only see their projection onto it. The shaded region has rank 3 and thus is inaccessible when of the form (4.1). The second row of plots in Figure 4.6 shows the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ produced by ALS. The horizontal reference lines are at 0, $\phi_1 = \phi_2$, and ϕ_3; eventually, $\boldsymbol{\alpha} \to \boldsymbol{\phi}$ and $\boldsymbol{\beta} \to \mathbf{0}$. The third row shows the scalars a and b, which have $\max_i\{\max(|a|, |b|)\} \approx 1.819$ and final values $(a, b) \approx (-0.371, 1.306)$; thus there is only moderate intermediate cancellation. The fourth row shows the corresponding errors and changes in errors.

A description of this swamp in our framework is complicated by lack of symmetry. Since $\boldsymbol{\phi} \neq \phi\mathbf{1}$, the set $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$ is not invariant, and so we cannot visualize by plots in (α, β) as in subsection 3.3. However, we find a qualitative match for Paatero's swamp by taking $\phi = \text{mean}(\boldsymbol{\phi}) \approx 0.260$. We plot this case in Figure 4.7 and observe the following:

- There is a local infimum at $\alpha = \beta \approx -\phi$, and valleys with large flow and algorithm time lead to it.
- There is a saddle-like feature at $\alpha = \beta \approx -\phi/3$ with sectors of both strong stability and strong instability near it.
- There are valleys with large flow and algorithm time leading to the solutions $(0, \phi)$ and $(\phi, 0)$. The valley hits the diagonal at $\alpha = \beta \approx \phi/3$ and does not appear to be steep-sided there. However, it has large flow and algorithm time there, indicating that the valley is steep-sided in another symmetry; we saw similar behavior in subsection 4.3.

Selecting $(\alpha_0, \beta_0) = (\text{mean}(\boldsymbol{\alpha}_0), \text{mean}(\boldsymbol{\beta}_0))$ leads to the local infimum at $\alpha = \beta \approx -\phi$. Selecting $(\alpha_0, \beta_0) = (\text{mean}(\boldsymbol{\alpha}_0), \text{mean}(\boldsymbol{\beta}_0))/3 \approx (-0.046, -0.228)$ produces a swamp quite similar to Paatero's swamp, and so we use that starting point.

Our explanation of the swamp is as follows and is illustrated in Figure 4.8:

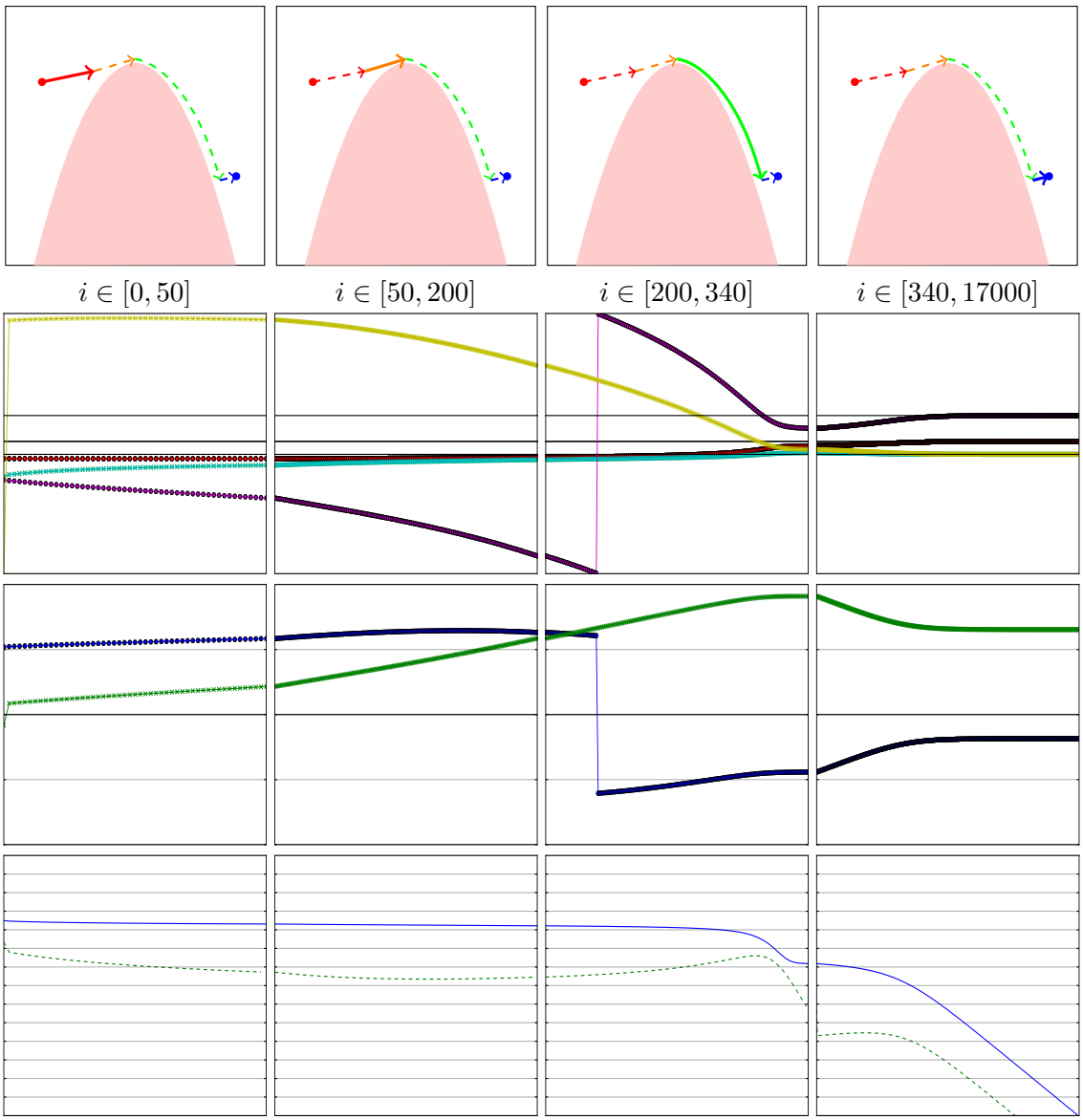

Figure 4.6. *Illustration of Paatero's swamp, using his perspective.*

1. The initial conditions are near a saddle-like feature on the $\boldsymbol{\alpha} = \boldsymbol{\beta}$ diagonal. The flow from this point would lead to the local infimum, but the flow has transverse instability (or weak enough stability that ALS can break free). ALS breaks the symmetry in one direction, maintaining the partial symmetry $\alpha_2 \approx \alpha_3$ and $\beta_2 \approx \beta_3$. Within Figure 4.8 we fix $\alpha_2 = \alpha_3 \approx -0.079$ and $\beta_2 = \beta_3 \approx -0.135$, which are their approximate values after 20 iterations, and plot the error as a function of (α_1, β_1). The flow decreases α_1 toward $\phi/2 - \pi/2$ and increases β_1 toward $\phi/2 + \pi/2$, which are the same point due to periodicity. This phase takes about 120 iterations.

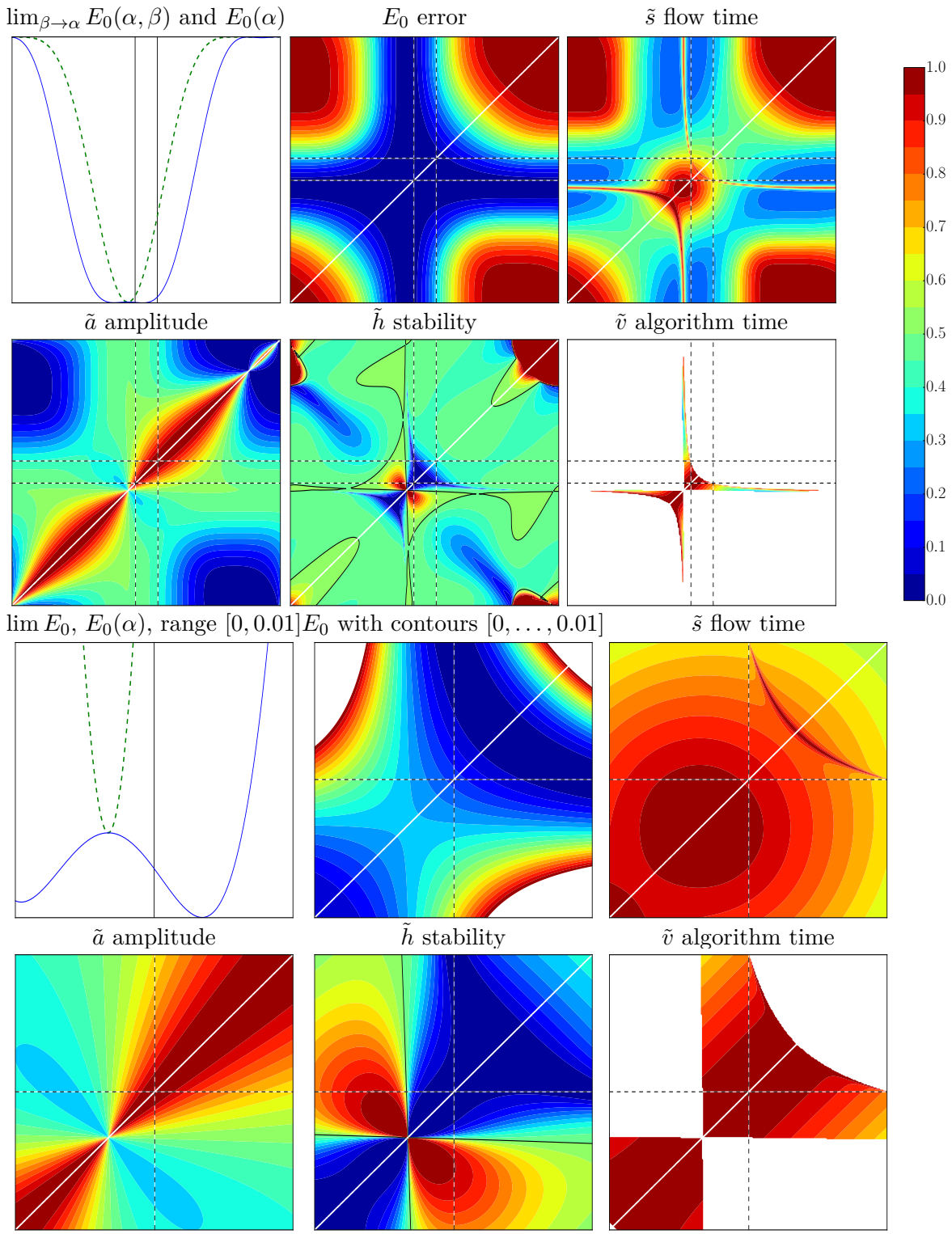

Figure 4.7. *The case $(d, \lambda) = (3, 0)$ and $(z, \phi) \approx (-0.284, 0.260)$, which is our symmetrized version of the parameters in Paatero's swamp example. The plots have symmetry $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$, and the bottom six plots are zoomed in to $(\alpha, \beta) \in [-\phi, \phi]^2$.*

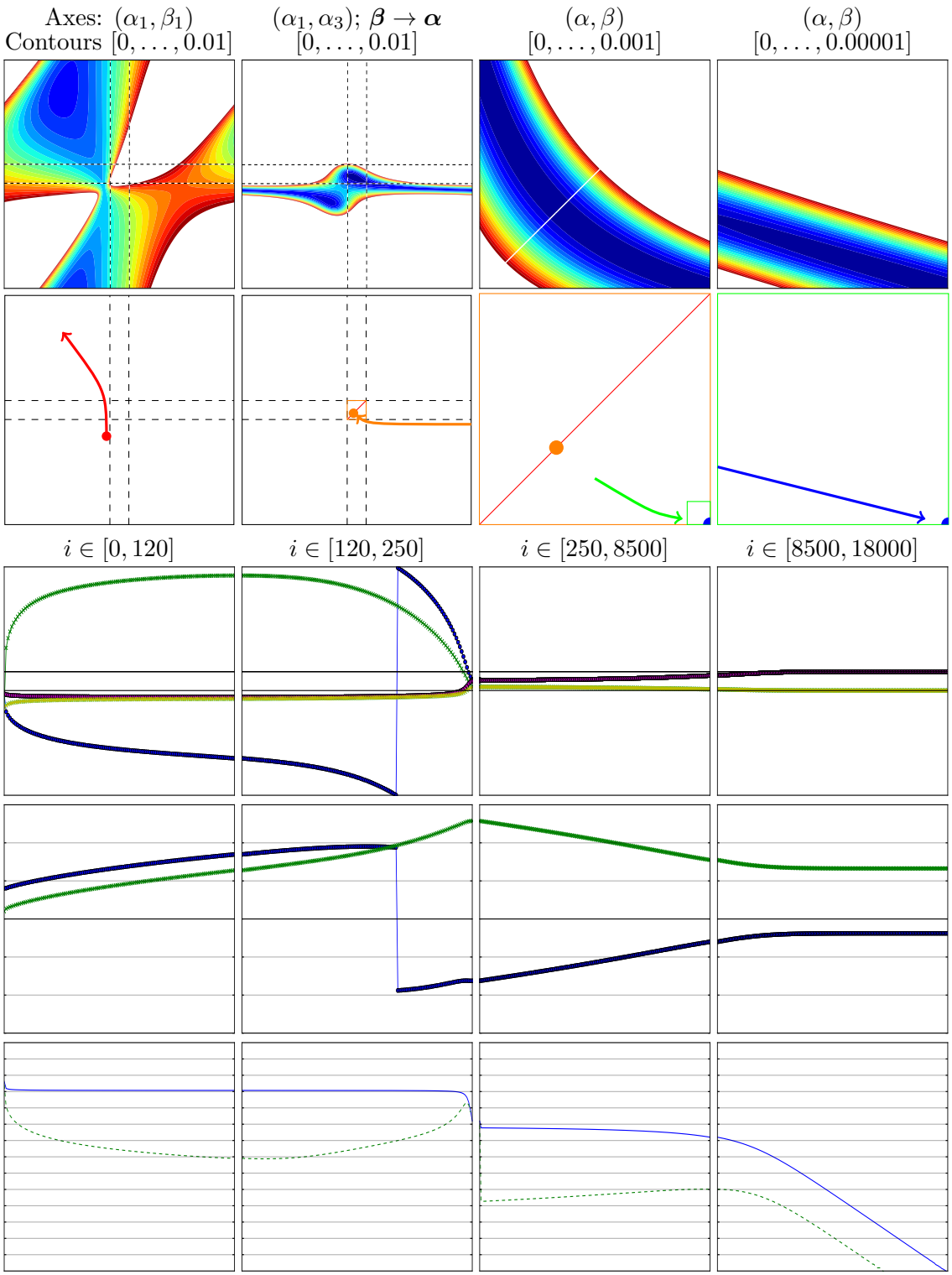

Figure 4.8. *Illustration of Paatero's swamp, using our perspective.*

2. The flow moves in the vicinity of (although not particularly close to) an essential saddle with $\boldsymbol{\alpha} = \boldsymbol{\beta}$ and $\alpha_2 = \alpha_3$. Descending from the essential saddle along a narrow valley, the algorithm moves slowly through a moderate transient swamp, which ends back in the symmetric set $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$. This phase takes about 130 iterations. (Compare to the third column in Figure 4.2.)
3. Next there is slow convergence within the symmetric set in a transient swamp caused by a valley descending from an essential saddle. This stage has a distinct start around iteration 250 but an indistinct end around 8500. The E_0 plot is from Figure 4.7, zoomed to $(\alpha, \beta) \in [0, \phi]^2$. (Compare to the middle column of Figure 4.5.)
4. Finally, there are many iterations of slow convergence in a terminal swamp due to an ill-conditioned Hessian at the solution with a condition number about 9427. The E_0 plot is from Figure 4.7, zoomed to $(\alpha, \beta) \in [9\phi/10, \phi] \times [0, \phi/10]$. (Compare to the last column of Figure 4.5.)

In comparing our explanation with Paatero's, we observe the following:

- The first three stages of Paatero's explanation match fairly well with the first two of ours. Both perspectives describe a pathway around a barrier.
- Paatero's fourth-stage explanation that progress is slow "because" the approximation possesses degeneracy is unfounded. Specifically, being near a degenerate tensor does not imply being in a transient swamp. In many places in Figure 3.2, $\tilde{a}$ is large, and so the tensor is nearly degenerate, but the flow time $\tilde{s}$ is small and the algorithm time $\tilde{v}$ is small or undefined, and so there is no transient swamp.
- Our third-stage explanation that progress is slow because it is in a valley descending from an essential saddle is justified by large flow time $\tilde{s}$ and algorithm time $\tilde{v}$, as well as the general behavior of algorithms in valleys. Note that the amplitude $\tilde{a}$ plots in Figure 4.7, as well as those in Figures 3.4, 4.3, and 4.4, show that as $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ moves up the valley toward the essential saddle, one has $(|a|, |b|) \to (\infty, \infty)$, which signals that the limiting tensor has rank greater than 2 and thus is degenerate. Thus tensors in a valley descending from an essential saddle are nearly degenerate; Paatero correctly observed a symptom of being in a transient swamp but incorrectly named it as the cause.
- Paatero misses our fourth stage of slow convergence due to an ill-conditioned Hessian.

Remark 4.1. The bound on the convergence rate of the Riemannian Gauss–Newton algorithm [6, Theorem 1] depends on the tensor condition number of [4, 5] and becomes worse as the condition number increases. Numerical experiments validated that both convergence [6] and transient progress [7] are slower when the condition number is large, which may lead one to believe that swamps are caused by a large condition number.

The condition number goes to infinity as a tensor representation approaches a degenerate tensor [5, Theorem 1.4] or a tensor with locally nonunique representation [5, Theorem 1.3]. As noted above, the up-valley limit toward an essential saddle leads to a degenerate tensor, but degenerate tensors also occur where there is no swamp. Moreover, ALS converges easily to a nonunique representation of the tensor generated by $\sin(\sum_{i=1}^{d} x_i)$ [2, 3, 29, 30]. Thus we conclude that a large condition number is a symptom of a swamp but not a cause.

4.5. Further swamps. In [17] we identified conditions that were likely to yield swamps for G_1 and numerically validated three of them. Further tests show that in all these cases the severity of the swamp is quite sensitive to the parameters:

- For $(z, \phi) = (1, \phi_0)$, a nonhyperbolic minimum at $\phi/2$ creates a terminal swamp with sublinear convergence. Setting $(d, \lambda) = (6, 0)$ and starting point $\boldsymbol{\alpha} = \phi\mathbf{1}$, it takes 524 iterations before the change in error hits 10^{-9}, 1129 iterations for 10^{-10}, 2433 for 10^{-11}, etc. (See [17, section 4.5].) Setting $\phi = 0.99\phi_0$, by 225 iterations the algorithm has converged with change in error under 10^{-14}; setting $\phi = 1.01\phi_0$ requires 127 iterations to converge.
- For $0 < z < 1$ and ϕ close to but less than a transcritical bifurcation value, a nearly nonhyperbolic saddle in the symmetric set creates a transient swamp for flows that pass near it. The profile of the change in error is similar to Figure 4.2, although the error itself never gets small. Fixing $(d, z, \lambda) = (6, 1/2, 0)$ and using starting point $\boldsymbol{\alpha} = \phi\mathbf{1}$, setting $\phi = 1.05503$ gives a transient swamp of length 760. (See [17, section 4.5].) Setting $\phi = 1.05502$ gives a transient swamp of length 345, $\phi = 1.055$ gives one of length 216, and $\phi = 1.05$ gives one of length 22. Setting $\phi = 1.0505$ exceeds the bifurcation value and gives a terminal swamp (to the local minimum) of length 325.
- For any $d > 2$ and $\phi \in (0, \pi/2)$, a specific z identified in [17, Lemma 17] produces a nonhyperbolic saddle with one positive and $d - 1$ zero eigenvalues located at $\boldsymbol{\alpha}_0 = (\phi/2 + \pi/4)\mathbf{1}$. Setting $(d, \phi, \lambda) = (6, \pi/8, 0)$ gives $z \approx -0.199$ and then starting at $\boldsymbol{\alpha} \approx \boldsymbol{\alpha}_0$ gives a transient swamp. (See [17, section 5.4.3].) The length of the swamp depends on how close $\boldsymbol{\alpha}$ is to $\boldsymbol{\alpha}_0$, with $\boldsymbol{\alpha} = 0.99\boldsymbol{\alpha}_0$ yielding length 15, $0.999\boldsymbol{\alpha}_0$ yielding length 100, $0.9999\boldsymbol{\alpha}_0$ yielding length 960, $1.0001\boldsymbol{\alpha}_0$ yielding length 7135, $1.001\boldsymbol{\alpha}_0$ yielding length 720, and $1.01\boldsymbol{\alpha}_0$ yielding length 75.

We thus believe these are *not* the swamps observed in the literature.

In the examples in subsections 4.1 to 4.3, the two terms in T were balanced in that $z = \pm 1$, and one should wonder whether any of the behavior is an artifact of this balance:

- In the terminal swamp example in subsection 4.1, if we set $z = 0.9$, then there is still a terminal swamp of length 270, and if we set $z = 0.5$, then there is one of length 300. Thus the terminal swamp is not an artifact.
- In the transient swamp example in subsection 4.2, if we set $z = 0.9$, then there is a transient swamp of length 610 followed by a terminal swamp of length 200. Thus the final rapid convergence is an artifact, but the transient swamp itself is not.
- In the double swamp in subsection 4.3, if we set $z = -0.9$, then the transient phase is 20 iterations longer, and if we set $z = -0.5$, then the transient phase is 250 longer; the terminal phase is unchanged. Thus neither is an artifact.

Similarly, one should wonder whether any of the behavior is an artifact of the symmetry $\boldsymbol{\phi} = \phi\mathbf{1}$. To test this, in the examples in subsections 4.1 to 4.3 we set $\boldsymbol{\phi} = \phi[0.9, 0.93, 0.97, 1.03, 1.07, 1.1]^*$. In all three examples the qualitative behavior stays the same. The terminal swamp length stays the same at 270, the transient swamp shortens slightly to 530, and the double swamp lengthens slightly to 616. We also note that the example in subsection 4.4 has $z \neq \pm 1$ and in its original formulation has $\boldsymbol{\phi} \neq \phi\mathbf{1}$.

4.5.1. Local infima. In Figure 4.3 we plotted the case $(d, z, \phi, \lambda) = (6, -1, \pi/8, 0)$ with symmetry $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha\mathbf{1}, \beta\mathbf{1})$. We can observe that there are local infima at $\alpha = \beta \approx -\phi$ and $\alpha = \beta \approx 2\phi$. The flow is stable near these infima, and both the flow time and the algorithm time are large there. Starting from $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (2\phi\mathbf{1}, 3\phi\mathbf{1})$, the iterations approach the local infimum at $\alpha = \beta \approx 2\phi$. The change in error decreases slowly at a decreasing rate.

5. Algorithmic implications. A primary motivation for the study of swamps is to enable the development of more effective algorithms for solving (1.1). Now that we have an understanding of the causes of swamps, and we can consider the following questions:

1. What algorithms or features of algorithms will produce faster convergence during terminal swamps?
2. What algorithms or features of algorithms will produce faster escape from transient swamps?
3. What algorithms or features of algorithms will avoid entering transient swamps?

We discuss these issues briefly in the following sections and consider the use of regularization in a little more detail.

5.1. Terminal swamps. The ill-conditioned Hessian at the solution that causes a terminal swamp can be treated by existing optimization methods. Algorithms to solve (1.1) have already been developed based on nonlinear conjugate gradient (NCG) (see, e.g., [1, 16, 33, 40]), ALS with NCG [14], ALS with line search (see, e.g., [10, 36, 39]), ALS with extrapolation (see, e.g., [8, 12, 19, 20, 24, 36, 49]), ALS with over-relaxation [27], algebraic multigrid [13], Newton's method (see, e.g., [15, 22, 35, 40, 48]), and Riemannian Gauss–Newton (see, e.g., [6, 7]). We have nothing to add to the discussion of these methods but do note that a good set of test cases can be generated by varying ϕ and d in the example of subsection 4.1.

5.2. Escaping a transient swamp. To escape from a transient swamp, the algorithm must be able to move rapidly down the steep-sided valley emanating from an essential saddle. In principle, the same list of methods adapted to an ill-conditioned Hessian at the solution should apply to this valley. However, an essential saddle is more like a nonhyperbolic saddle (i.e., singular Hessian) and thus will still be difficult for these algorithms. Without making any claim regarding these methods, we will instead discuss another option.

The analysis in section 3 and Appendix A applies for regularization $\lambda \geq 0$, but the example swamps in section 4 all set $\lambda = 0$. Setting $\lambda > 0$ makes the minimization problem well posed (see, e.g., [29]) and the error function continuous (see Lemma A.1), thus eliminating essential saddles. In Figure 5.1 we visualize the case with $(d, z, \phi) = (6, 1, \pi/8)$ under the constraints $\alpha_2 = \cdots = \alpha_d$ and $\boldsymbol{\beta} = \phi - \boldsymbol{\alpha}$ as in Figure 3.4 but set $\lambda = 1/10$. We observe that the point of the steep valley emanating from $(\alpha_1, \alpha_d) = (\phi/2 - \pi/2, \phi/2)$ has been rounded off, thereby removing the worst part of the transient swamp. The minimum points are no longer at the solutions but are close enough that ALS with $\lambda = 0$ would converge rapidly from there.

This suggests modifying the ALS algorithm by turning on regularization whenever a transient swamp is encountered. To detect when the iteration is in a transient swamp, one can look for periods when the change (improvement) in error is increasing, or equivalently when the error is concave down. This criterion was used as the definition of a (transient) swamp in [28].

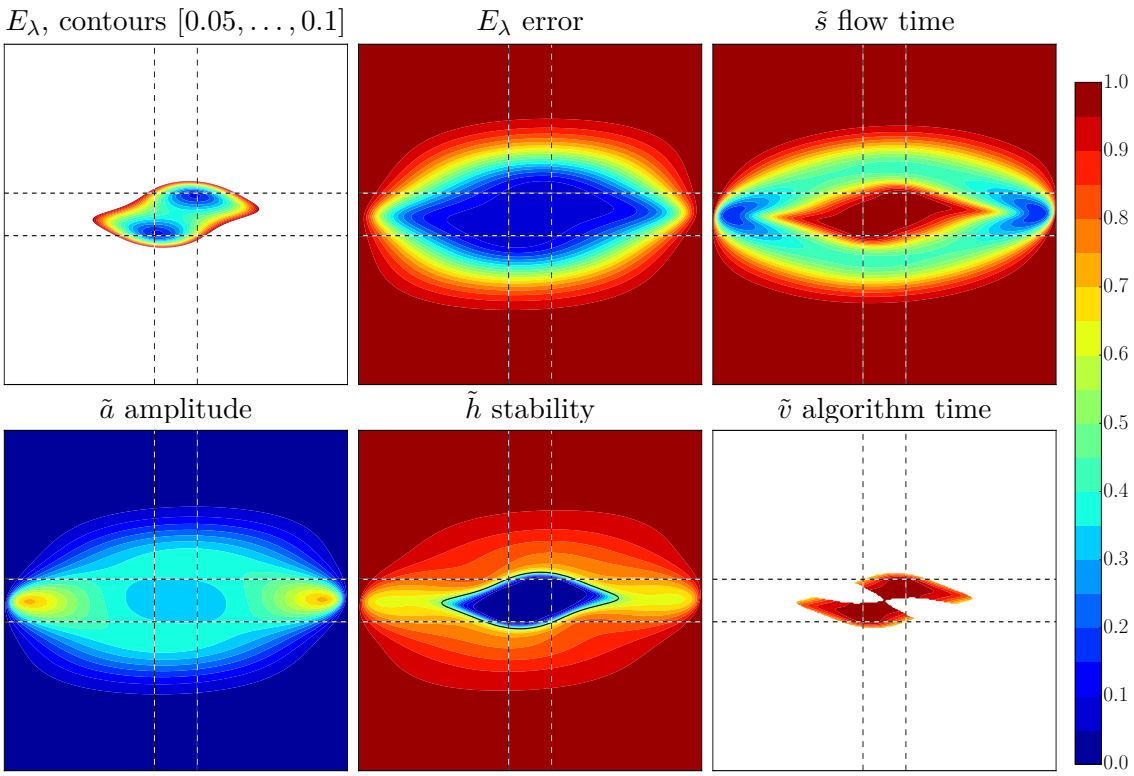

Figure 5.1. *Visualization of the case* $(d, z, \phi, \lambda) = (6, 1, \pi/8, 1/10)$ *under the constraints* $\alpha_2 = \cdots = \alpha_d$ *and* $\boldsymbol{\beta} = \phi - \boldsymbol{\alpha}$*, with horizontal axis* α_1 *and vertical axis* α_d*. Compare with the* $\lambda = 0$ *case in Figure 3.4 to see how the steep-sided valley caused by the essential singularity has been removed.*

To test this idea, we consider the transient swamp in subsection 4.2 and the transient portion of the double swamp in subsection 4.3. We first run ALS iterations with $\lambda = 0$ until the change in the error E_0 hits a minimum. We then set λ to various values and continue with ALS, noting how many further iterations it takes until the change in the error E_λ hits a maximum. The results are given in Table 5.1. We see that too small λ has no effect, but large enough λ provides an effective way to exit the swamp. In all cases the $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ upon escape was similar enough to the $\lambda = 0$ case that the final convergence was similar. For the subsection 4.2 example, at larger λ there was some degradation in final convergence speed because the symmetry $\boldsymbol{\beta} = \phi - \boldsymbol{\alpha}$ was not as well maintained. For the subsection 4.3 example, by $\lambda = 10^{-3}$ the change in error E_λ was strictly decreasing and the transient swamp had merged into the terminal swamp. Thus we conclude that adding regularization can be an effective way to escape from a transient swamp.

Regularization, in various forms, has previously been used to alleviate swamps. In [37], if the matrix in the normal equations used for ALS updates was ill-conditioned, then its smallest eigenvalue was increased. In [32, 34], a regularized error function of the form

$$\|T - G\|^2 + \lambda \sum_{i=1}^{d} \left\| \begin{bmatrix} G_i^1 & G_i^2 & \ldots & G_i^r \end{bmatrix} \right\|^2 \tag{5.1}$$

Table 5.1
The effect of regularization on escaping from a transient swamp.

Test case	Initial iterations	Iterations to escape using $\lambda = \cdots$							
		0	10^{-11}	10^{-9}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}
subsection 4.2	26	573	573	571	427	207	76	26	7
subsection 4.3	7	285	285	285	285	281	249	131	0

was used within a Gauss–Newton algorithm. Although not equivalent to the regularization we use, they are expected to have similar effects. We now know why such regularization works.

Remark 5.1. The "regularization" in [26, 31] is of a fundamentally different form than that used here. Its effect is to make the method proximal (see, e.g., [38]), which should slow the escape from a transient swamp.

5.3. Avoiding a transient swamp. The motivation to avoid transient swamps is strongly affected by how difficult they are to escape from. If simply adding regularization proves to be an effective way to escape in general, then one may decide that the path by an essential saddle is a good one to take.

Setting $\lambda > 0$ can also be used to keep the flow from getting near the location of the essential saddle in the first place. In [34, Figure 3], the example of subsection 4.4 was modified by including regularization using the method from [37] for the first 120 iterations. Using regularization in our form with $\lambda = 10^{-2}$ for the first 120 iterations, we find that the flow takes a fundamentally different path through $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-space, skips the two transient swamps entirely, and is left with only the terminal portion. Similarly, regularization with $\lambda = 10^{-2}$ for the first 10 iterations in the double swamp example in subsection 4.3 causes the iteration to approach $(\alpha, \beta) = (0, \phi)$ from $\alpha < 0 < \phi < \beta$ rather than $0 < \alpha < \beta < \phi$ and skip the transient swamp entirely.

In principle, since the essential saddle is a feature of the error function itself, algorithms that descend more rapidly than ALS should pass nearer to the essential saddle and thus go more deeply into the transient swamp. However, there are two additional effects, which were hidden in our analysis, that may sometimes keep non-ALS algorithms away from essential saddles:

- As noted in Remark 3.2, the ALS algorithm automatically maintains (a, b) at their optimal values and thus justifies making them fast variables, as described in subsection 3.1. Non-ALS algorithms do not have this property, and so (a, b) will generally not have their optimal values, and thus the error landscapes will differ from those we studied and plotted. We expect the deviation to be greatest when (a, b) change rapidly, which is exactly what we observe in Figure 4.2 (although not in Figure 4.5) when the iteration moves near the essential saddle.
- In our formulation of the model problem, any rank-2 target can be put into the standard form (1.2) if we extend the vectors $[\ 1 \quad 0\ ]^*$ and $[\ \cos(\phi_i) \quad \sin(\phi_i)\ ]^*$ with zeros up to the original sizes of T_i^l. Under our assumption that $G_i^l \in \operatorname{span}\{T_i^1, T_i^2\}$, the vectors in $G_1 =$ (1.3) and $G_2 =$ (1.4) are also extended by zeros and these additional entries have no effect. Due to its action as an alternating orthogonal projection [29, section 4.1], the ALS algorithm enforces the assumption $G_i^l \in \operatorname{span}\{T_i^1, T_i^2\}$, and so our

model is valid for it. Non-ALS algorithms do not have this projection property, and so they do not enforce $G_i^l \in \operatorname{span}\{T_i^1, T_i^2\}$, and the vectors in $G_1 =$ (1.3) and $G_2 =$ (1.4) are generally extended with nonzeros. These nonzero entries add to the error and will eventually be suppressed by the algorithm. In the short run, however, they affect the error landscape and may be acting as a regularization of the error that ALS sees. (This effect might also explain the swamp-avoiding feature of the algorithm in [26, 31].)

Appendix A. Analysis of the model problem of fitting a rank-2 tensor with a rank-2 tensor. In this section we analyze the model problem of fitting the rank-2 target T in (1.2) by the rank-2 tensor G_2 in (1.4), as sketched in section 3. Appendix A.1 establishes notation and other preliminaries. In Appendix A.2 we describe how to make the scalar coefficients (a, b) into fast variables. In Appendix A.3 we determine how the error, gradient, and Hessian depend on these angular variables. In Appendix A.4 we analyze the behavior of the approximation on the diagonal $\boldsymbol{\beta} = \boldsymbol{\alpha}$ when $\lambda > 0$ and in the limit $\boldsymbol{\beta} \to \boldsymbol{\alpha}$ when $\lambda = 0$. In Appendix A.5 we analyze the case with symmetric T and G_2.

A.1. Notation and other preliminaries. We denote vectors with brackets as in $\left[\begin{smallmatrix} x \\ -y \end{smallmatrix}\right]$. Items in parentheses are merely grouped even if stacked, and so $\left(\begin{smallmatrix} x \\ -y \end{smallmatrix}\right) = x - y$. We will use a nonstandard version of the quotient rule

$$\left(\frac{u}{v}\right)' = \frac{u' - \left(\frac{u}{v}\right) v'}{v}\,, \tag{A.1}$$

which allows us to refer to previous functions rather than rewriting them.

We let $\mathbf{1}_k$ be the (column) vector of size k with entries 1, I_k be the identity matrix of size $k \times k$, and $\mathbf{e}_j$ be the vector with 1 in entry j and otherwise 0. For $1 \le q \le p$, let the matrix

$$U_{pq} = I_p - 2\frac{(\mathbf{e}_q - \mathbf{1}_p/\sqrt{p})(\mathbf{e}_q - \mathbf{1}_p/\sqrt{p})^*}{(\mathbf{e}_q - \mathbf{1}_p/\sqrt{p})^*(\mathbf{e}_q - \mathbf{1}_p/\sqrt{p})} \tag{A.2}$$

be the Householder reflector [21] that takes $\mathbf{1}_p$ to $\sqrt{p}\mathbf{e}_q$; it is unitary and is its own conjugate transpose and inverse.

For any two (real-valued) tensors F and G indexed by $j_i = 1, 2, \dots, M_i$ for $i = 1, \dots d$, define the inner product by

$$\langle F, G\rangle = \sum_{j_1=1}^{M_1} \cdots \sum_{j_d=1}^{M_d} F(j_1, \dots, j_d) G(j_1, \dots, j_d) \tag{A.3}$$

and the norm by $\|F\| = \sqrt{\langle F, F\rangle}$.

We define

$$p(\boldsymbol{\alpha}) = \left\langle \bigotimes_{i=1}^{d} \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \bigotimes_{i=1}^{d} \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix} \right\rangle = \prod_{i=1}^{d} \cos(\alpha_i)\,, \tag{A.4}$$

which is the inner product of normalized rank-1 tensors whose factors have angles $\mathbf{0}$ and $\boldsymbol{\alpha}$. If the tensors have angles $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, then their inner product is $p(\boldsymbol{\alpha} - \boldsymbol{\beta})$. We define

$$n(\boldsymbol{\alpha}) = \left\langle \bigotimes_{i=1}^{d} \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix}, T \right\rangle = \frac{p(\boldsymbol{\alpha}) + zp(\boldsymbol{\alpha} - \boldsymbol{\phi})}{(1 + 2zp(\boldsymbol{\phi}) + z^2)^{1/2}}\,, \tag{A.5}$$

which is the inner product of a normalized rank-1 tensor with the target. We denote the partial derivatives of these functions by

$$p_j(\boldsymbol{\alpha}) = \frac{\partial}{\partial \alpha_j} p(\boldsymbol{\alpha}) = -\sin(\alpha_j) \prod_{i=1,\neq j}^{d} \cos(\alpha_i), \tag{A.6}$$

$$n_j(\boldsymbol{\alpha}) = \frac{\partial}{\partial \alpha_j} n(\boldsymbol{\alpha}) = \frac{p_j(\boldsymbol{\alpha}) + z p_j(\boldsymbol{\alpha} - \boldsymbol{\phi})}{(1 + 2zp(\boldsymbol{\phi}) + z^2)^{1/2}}\,. \tag{A.7}$$

When $j \neq k$, the mixed partial derivatives are

$$p_{jk}(\boldsymbol{\alpha}) = \frac{\partial}{\partial \alpha_k} p_j(\boldsymbol{\alpha}) = \sin(\alpha_j)\sin(\alpha_k) \prod_{i=1,\neq j,k}^{d} \cos(\alpha_i), \tag{A.8}$$

$$n_{jk}(\boldsymbol{\alpha}) = \frac{\partial}{\partial \alpha_k} n_j(\boldsymbol{\alpha}) = \frac{p_{jk}(\boldsymbol{\alpha}) + z p_{jk}(\boldsymbol{\alpha} - \boldsymbol{\phi})}{(1 + 2zp(\boldsymbol{\phi}) + z^2)^{1/2}}\,. \tag{A.9}$$

When $j = k$ we have

$$p_{jj}(\boldsymbol{\alpha}) = \frac{\partial}{\partial \alpha_j} p_j(\boldsymbol{\alpha}) = -p(\boldsymbol{\alpha}), \tag{A.10}$$

$$n_{jj}(\boldsymbol{\alpha}) = \frac{\partial}{\partial \alpha_j} n_j(\boldsymbol{\alpha}) = -n(\boldsymbol{\alpha})\,. \tag{A.11}$$

A.2. Making the scalars into fast variables. As discussed in subsection 3.1, the dependence of $E_\lambda(a, \boldsymbol{\alpha}, b, \boldsymbol{\beta})$ on (a, b) can be removed by making them always assume the values that minimize $E_\lambda(a, \boldsymbol{\alpha}, b, \boldsymbol{\beta})$. To do so, one poses the least-squares problem (3.2), sets up the normal equations (3.3), and then solves for (a, b). With the notation introduced above, the normal equations (3.3) become

$$\begin{bmatrix} 1+\lambda & p(\boldsymbol{\alpha}-\boldsymbol{\beta}) \\ p(\boldsymbol{\alpha}-\boldsymbol{\beta}) & 1+\lambda \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} n(\boldsymbol{\alpha}) \\ n(\boldsymbol{\beta}) \end{bmatrix}\,. \tag{A.12}$$

By the general theory of least-squares fitting, when $\lambda = 0$ the matrix in (A.12) is always positive semidefinite and is positive definite (and hence invertible) if and only if the two separable terms in G_2 are linearly independent. When $\lambda > 0$, the matrix is always positive definite since including $\lambda > 0$ adds a positive multiple of the identity to a positive semidefinite matrix. The coefficients can be determined by inverting the matrix (using Cramer's rule) to obtain

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{\begin{bmatrix} n(\boldsymbol{\alpha})(1+\lambda) - n(\boldsymbol{\beta})p(\boldsymbol{\alpha}-\boldsymbol{\beta}) \\ n(\boldsymbol{\beta})(1+\lambda) - n(\boldsymbol{\alpha})p(\boldsymbol{\alpha}-\boldsymbol{\beta}) \end{bmatrix}}{(1+\lambda)^2 - p^2(\boldsymbol{\alpha}-\boldsymbol{\beta})}\,. \tag{A.13}$$

A.3. Error, gradient, and Hessian as a function of the angles. Evaluating directly, we have

$$\|G_2\|^2 + \lambda(a^2 + b^2) = a\left((1+\lambda)a + bp(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) + b\left((1+\lambda)b + ap(\boldsymbol{\alpha}-\boldsymbol{\beta})\right)\,. \tag{A.14}$$

Using (A.12) and (A.13) and rearranging, we then have

$$\begin{aligned}\|G_2\|^2 + \lambda(a^2+b^2) &= an(\boldsymbol{\alpha}) + bn(\boldsymbol{\beta}) \\ &= \frac{(1+\lambda)\left(n^2(\boldsymbol{\alpha}) + n^2(\boldsymbol{\beta})\right) - 2n(\boldsymbol{\alpha})n(\boldsymbol{\beta})p(\boldsymbol{\alpha}-\boldsymbol{\beta})}{(1+\lambda)^2 - p^2(\boldsymbol{\alpha}-\boldsymbol{\beta})} = A(\boldsymbol{\alpha},\boldsymbol{\beta})\,,\end{aligned} \tag{A.15}$$

where the notation $A(\boldsymbol{\alpha},\boldsymbol{\beta})$ is introduced since (A.15) will occur in many later formulas. By making the scalar variables fast as in Appendix A.2 we eliminated a and b. Since the linear least-squares fitting is an orthogonal projection, we then know that

$$E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) = \|T\|^2 - \left(\|G_2\|^2 + \lambda\left(a^2+b^2\right)\right) = 1 - A(\boldsymbol{\alpha},\boldsymbol{\beta})\,. \tag{A.16}$$

Using (A.1), we have

$$\frac{\partial}{\partial\alpha_j}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{\begin{pmatrix} -2(1+\lambda)n(\boldsymbol{\alpha})n_j(\boldsymbol{\alpha}) \\ +2n(\boldsymbol{\beta})\left(n_j(\boldsymbol{\alpha})p(\boldsymbol{\alpha}-\boldsymbol{\beta}) + n(\boldsymbol{\alpha})p_j(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) \\ -2A(\boldsymbol{\alpha},\boldsymbol{\beta})p(\boldsymbol{\alpha}-\boldsymbol{\beta})p_j(\boldsymbol{\alpha}-\boldsymbol{\beta}) \end{pmatrix}}{(1+\lambda)^2 - p^2(\boldsymbol{\alpha}-\boldsymbol{\beta})}\,. \tag{A.17}$$

The partial derivative with respect to β_j is obtained by permuting the variables (noticing that while $p(\boldsymbol{\alpha}-\boldsymbol{\beta})$ is even, $p_j(\boldsymbol{\alpha}-\boldsymbol{\beta})$ is odd). The gradient has two blocks, corresponding to the α and β variables.

The Hessian has a 2×2 block structure, with blocks corresponding to the α and β variables. When $j=k$, some of the terms combine and we can use (A.10) and (A.11) to simplify slightly. On the diagonal blocks are entries of the form

$$\begin{aligned}&\frac{\partial^2}{\partial\alpha_j\partial\alpha_k}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ &= 2\begin{pmatrix} -(1+\lambda)\left(n_k(\boldsymbol{\alpha})n_j(\boldsymbol{\alpha}) + n(\boldsymbol{\alpha})n_{jk}(\boldsymbol{\alpha})\right) \\ +n(\boldsymbol{\beta})\left(n_{jk}(\boldsymbol{\alpha})p(\boldsymbol{\alpha}-\boldsymbol{\beta}) + n_j(\boldsymbol{\alpha})p_k(\boldsymbol{\alpha}-\boldsymbol{\beta}) + n_k(\boldsymbol{\alpha})p_j(\boldsymbol{\alpha}-\boldsymbol{\beta}) + n(\boldsymbol{\alpha})p_{jk}(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) \\ +p(\boldsymbol{\alpha}-\boldsymbol{\beta})\left(p_k(\boldsymbol{\alpha}-\boldsymbol{\beta})\frac{\partial}{\partial\alpha_j}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) + p_j(\boldsymbol{\alpha}-\boldsymbol{\beta})\frac{\partial}{\partial\alpha_k}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta})\right) \\ -A(\boldsymbol{\alpha},\boldsymbol{\beta})\left(p_k(\boldsymbol{\alpha}-\boldsymbol{\beta})p_j(\boldsymbol{\alpha}-\boldsymbol{\beta}) + p(\boldsymbol{\alpha}-\boldsymbol{\beta})p_{jk}(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) \end{pmatrix} \\ &= \frac{}{(1+\lambda)^2 - p^2(\boldsymbol{\alpha}-\boldsymbol{\beta})}\,.\end{aligned} \tag{A.18}$$

On the off-diagonal blocks are entries of the form

$$\begin{aligned}&\frac{\partial^2}{\partial\alpha_j\partial\beta_k}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ &\qquad = 2\begin{pmatrix} n_k(\boldsymbol{\beta})\left(n_j(\boldsymbol{\alpha})p(\boldsymbol{\alpha}-\boldsymbol{\beta}) + n(\boldsymbol{\alpha})p_j(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) \\ -n(\boldsymbol{\beta})\left(n_j(\boldsymbol{\alpha})p_k(\boldsymbol{\alpha}-\boldsymbol{\beta}) + n(\boldsymbol{\alpha})p_{jk}(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) \\ -p(\boldsymbol{\alpha}-\boldsymbol{\beta})\left(p_k(\boldsymbol{\alpha}-\boldsymbol{\beta})\frac{\partial}{\partial\alpha_j}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) - p_j(\boldsymbol{\alpha}-\boldsymbol{\beta})\frac{\partial}{\partial\beta_k}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta})\right) \\ +A(\boldsymbol{\alpha},\boldsymbol{\beta})\left(p_k(\boldsymbol{\alpha}-\boldsymbol{\beta})p_j(\boldsymbol{\alpha}-\boldsymbol{\beta}) + p(\boldsymbol{\alpha}-\boldsymbol{\beta})p_{jk}(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) \end{pmatrix} \\ &\qquad = \frac{}{(1+\lambda)^2 - p^2(\boldsymbol{\alpha}-\boldsymbol{\beta})}\,.\end{aligned} \tag{A.19}$$

A.4. Behavior near the diagonal $\beta = \alpha$. If $\lambda > 0$, then we can simply set $\boldsymbol{\beta} = \boldsymbol{\alpha}$. In Lemma A.1, we show that G_2 effectively collapses to G_1 when $\lambda > 0$ and $\boldsymbol{\beta} = \boldsymbol{\alpha}$.

If $\lambda = 0$ and we set $\boldsymbol{\beta} = \boldsymbol{\alpha}$, then the denominators in (A.15) and (A.17)–(A.19) are zero. We therefore instead consider limits as $\boldsymbol{\beta} \to \boldsymbol{\alpha}$, which will depend on the direction from which $\boldsymbol{\beta} \to \boldsymbol{\alpha}$. In Lemma A.2 we compute the limit of the error both as a function of the direction of approach and as the minimum over all directions. In Lemma A.3 we show that the vector of scalars $[a, b]$ naturally decomposes into a converging, coalescing part and a diverging, canceling part. (It is also possible to use these methods to compute the limit of G_2 itself, but we do not need the result here.)

Lemma A.1. *If $\lambda > 0$ and we set $\boldsymbol{\beta} = \boldsymbol{\alpha}$, then the two terms in G_2 are identical and each is half of the single G_1 term obtained using $E_{\lambda/2}$. In the limit as $\lambda \to 0^+$, each of the two terms is half of the single G_1 term obtained using E_0.*

Proof. Since $n(\boldsymbol{\alpha}) = n(\boldsymbol{\beta})$, $p(\boldsymbol{\alpha} - \boldsymbol{\beta}) = 1$, and $p_j(\boldsymbol{\alpha} - \boldsymbol{\beta}) = 0$, we have

$$\text{(A.13)} \mapsto \qquad \begin{bmatrix} a \\ b \end{bmatrix} = \frac{n(\boldsymbol{\alpha})}{1 + \lambda/2} \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, \tag{A.20}$$

$$\text{(A.16)} \mapsto \qquad E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1 - \frac{n^2(\boldsymbol{\alpha})}{1 + \lambda/2}. \tag{A.21}$$

From [17], the optimal scalar for G_1 using regularization $\lambda/2$ is $a = n(\boldsymbol{\alpha})/(1 + \lambda/2)$ and (A.21) $= E_{\lambda/2}(\alpha)$. The formulas involving G_1 are all continuous at $\lambda = 0$ and thus yield the limit of the terms of G_2 as $\lambda \to 0^+$. ∎

Lemma A.2. *If $\lambda = 0$ and $\boldsymbol{\beta} = \boldsymbol{\alpha} + \epsilon \mathbf{v}$, with $\|\mathbf{v}\| = 1$, then*

$$\lim_{\epsilon \to 0} E_0(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1 - \left(n^2(\boldsymbol{\alpha}) + \left(\sum_{i=1}^{d} v_i n_i(\boldsymbol{\alpha}) \right)^2 \right), \tag{A.22}$$

which has a minimum value with respect to $\mathbf{v}$ of

$$1 - \left(n^2(\boldsymbol{\alpha}) + \sum_{i=1}^{d} n_i^2(\boldsymbol{\alpha}) \right). \tag{A.23}$$

Proof. Starting from (A.15) and applying L'Hôpital's rule twice, we obtain

$$\begin{aligned} \lim_{\epsilon \to 0} \|G_2\|^2 &\overset{\text{L'H}}{=} \lim_{\epsilon \to 0} \frac{\displaystyle\sum_{i=1}^{d} v_i \begin{pmatrix} 2n(\boldsymbol{\alpha} + \epsilon\mathbf{v}) n_i(\boldsymbol{\alpha} + \epsilon\mathbf{v}) \\ -2n(\boldsymbol{\alpha}) n_i(\boldsymbol{\alpha} + \epsilon\mathbf{v}) p(\epsilon\mathbf{v}) \\ +2n(\boldsymbol{\alpha}) n(\boldsymbol{\alpha} + \epsilon\mathbf{v}) p_i(\epsilon\mathbf{v}) \end{pmatrix}}{-\sum_{j=1}^{d} v_j 2p(\epsilon\mathbf{v}) p_j(\epsilon\mathbf{v})} \\ &= n^2(\boldsymbol{\alpha}) + \left(\sum_{i=1}^{d} v_i n_i(\boldsymbol{\alpha}) \right) \lim_{\epsilon \to 0} \frac{n(\boldsymbol{\alpha} + \epsilon\mathbf{v}) - n(\boldsymbol{\alpha}) p(\epsilon\mathbf{v})}{-\sum_{j=1}^{d} v_j p_j(\epsilon\mathbf{v})} \\ &\overset{\text{L'H}}{=} n^2(\boldsymbol{\alpha}) + \left(\sum_{i=1}^{d} v_i n_i(\boldsymbol{\alpha}) \right) \lim_{\epsilon \to 0} \frac{\sum_{i=1}^{d} v_i \left(n_i(\boldsymbol{\alpha} + \epsilon\mathbf{v}) + n(\boldsymbol{\alpha}) p_i(\epsilon\mathbf{v}) \right)}{-\sum_{j=1}^{d} \sum_{k=1}^{d} v_j v_k p_{jk}(\epsilon\mathbf{v})}, \end{aligned} \tag{A.24}$$

which yields (A.22). The minimum of (A.22) is achieved by setting $v_i = n_i(\boldsymbol{\alpha})$ and then normalizing $\mathbf{v}$. ∎

Lemma A.3. *If $\lambda = 0$ and $\boldsymbol{\beta} = \boldsymbol{\alpha} + \epsilon\mathbf{v}$, with $\|\mathbf{v}\| = 1$, then*

$$\lim_{\epsilon\to 0}\begin{bmatrix}a\\ b\end{bmatrix} = \frac{n(\boldsymbol{\alpha})}{2}\begin{bmatrix}1\\ 1\end{bmatrix} + \frac{1}{2}\left(\lim_{\epsilon\to 0}\frac{\sum_{i=1}^d v_i n_i(\boldsymbol{\alpha}+\epsilon\mathbf{v})}{\sum_{j=1}^d v_j p_j(\epsilon\mathbf{v})}\right)\begin{bmatrix}1\\ -1\end{bmatrix}, \tag{A.25}$$

which diverges except possibly when $\sum_{i=1}^d v_i n_i(\boldsymbol{\alpha}) = 0$. Choosing $\mathbf{v}$ to give the minimal error (A.23) results in

$$\lim_{\epsilon\to 0}\begin{bmatrix}a\\ b\end{bmatrix} = \frac{n(\boldsymbol{\alpha})}{2}\begin{bmatrix}1\\ 1\end{bmatrix} + \frac{1}{2}\left(\lim_{\epsilon\to 0}\frac{\sum_{i=1}^d n_i(\boldsymbol{\alpha}) n_i(\boldsymbol{\alpha}+\epsilon\mathbf{v})}{\sum_{j=1}^d n_j(\boldsymbol{\alpha}) p_j(\epsilon\mathbf{v})}\right)\begin{bmatrix}1\\ -1\end{bmatrix}, \tag{A.26}$$

which diverges except possibly when $\sum_{i=1}^d n_i^2(\boldsymbol{\alpha}) = 0$.

Proof. Starting with (A.13), applying L'Hôpital's rule, and collecting terms, we obtain

$$\begin{aligned}\lim_{\epsilon\to 0}\begin{bmatrix}a\\ b\end{bmatrix} &\overset{\text{L'H}}{=} \lim_{\epsilon\to 0}\frac{\sum_{i=1}^d v_i\begin{bmatrix}-n_i(\boldsymbol{\alpha}+\epsilon\mathbf{v})p(\epsilon\mathbf{v}) - n(\boldsymbol{\alpha}+\epsilon\mathbf{v})p_i(\epsilon\mathbf{v})\\ n_i(\boldsymbol{\alpha}+\epsilon\mathbf{v}) - n(\boldsymbol{\alpha})p_i(\epsilon\mathbf{v})\end{bmatrix}}{-\sum_{j=1}^d v_j 2p(\epsilon\mathbf{v})p_j(\epsilon\mathbf{v})}\\ &= \lim_{\epsilon\to 0}\frac{-\sum_{i=1}^d v_i p_i(\epsilon\mathbf{v})\begin{bmatrix}n(\boldsymbol{\alpha}+\epsilon\mathbf{v})\\ n(\boldsymbol{\alpha})\end{bmatrix}}{\left(-\sum_{j=1}^d v_j p_j(\epsilon\mathbf{v})\right)2p(\epsilon\mathbf{v})} + \lim_{\epsilon\to 0}\frac{\sum_{i=1}^d v_i n_i(\boldsymbol{\alpha}+\epsilon\mathbf{v})\begin{bmatrix}-p(\epsilon\mathbf{v})\\ 1\end{bmatrix}}{-\sum_{j=1}^d v_j 2p(\epsilon\mathbf{v})p_j(\epsilon\mathbf{v})}.\end{aligned} \tag{A.27}$$

Canceling common factors and evaluating the convergent parts of the limit yields (A.25). ∎

A.5. Analysis with symmetric T and G_2. In this section we assume $\boldsymbol{\phi} = \phi\mathbf{1}$, $\boldsymbol{\alpha} = \alpha\mathbf{1}$, and $\boldsymbol{\beta} = \beta\mathbf{1}$, which make T and G_2 symmetric under permutations of directions. If $\phi_j = \phi_k$, $\alpha_j = \alpha_k$, and $\beta_j = \beta_k$, then $\frac{\partial}{\partial\alpha_j}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{\partial}{\partial\alpha_k}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta})$ and $\frac{\partial}{\partial\beta_j}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{\partial}{\partial\beta_k}E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta})$, and so the equalities of the variables are preserved under the gradient flow. Thus the states with such symmetries are invariant sets for the gradient flow.

A.5.1. Decomposition of the Hessian. In this section we decompose the Hessian into a part transverse to the symmetric set where $\boldsymbol{\alpha} = \alpha\mathbf{1}$ and $\boldsymbol{\beta} = \beta\mathbf{1}$ and a part within the symmetric set. The transverse Hessian $H^\perp$ is constructed in Lemma A.4 using the procedure in subsection 2.1. The in-plane Hessian $H^\parallel$ is constructed in Lemma A.5 as the complement of $H^\perp$. We use both Hessians in Appendix A.5.2 to define two types of transverse stability and in Appendix A.5.3 to analyze the eigenvalues of the Hessian at the minimum.

Lemma A.4. *The transverse Hessian $H^\perp$ with respect to the invariant set $(\boldsymbol{\alpha},\boldsymbol{\beta}) = (\alpha\mathbf{1},\beta\mathbf{1})$*

can be written as $\hat{H}^{\perp} \otimes I_{d-1}$ *where the* 2×2 *symmetric matrix* $\hat{H}^{\perp}$ *has entries*

$$\hat{H}^{\perp}(1,1) = \frac{2\begin{pmatrix} (n(\boldsymbol{\alpha}) + n_{jk}(\boldsymbol{\alpha}))\left((1+\lambda)n(\boldsymbol{\alpha}) - n(\boldsymbol{\beta})p(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) \\ -(p(\boldsymbol{\alpha}-\boldsymbol{\beta}) + p_{jk}(\boldsymbol{\alpha}-\boldsymbol{\beta}))\left(n(\boldsymbol{\alpha})n(\boldsymbol{\beta}) - A(\boldsymbol{\alpha},\boldsymbol{\beta})p(\boldsymbol{\alpha}-\boldsymbol{\beta})\right) \end{pmatrix}}{(1+\lambda)^2 - p^2(\boldsymbol{\alpha}-\boldsymbol{\beta})}, \tag{A.28}$$

$$\hat{H}^{\perp}(1,2) = \frac{2\left(p(\boldsymbol{\alpha}-\boldsymbol{\beta}) + p_{jk}(\boldsymbol{\alpha}-\boldsymbol{\beta})\right)\left(n(\boldsymbol{\alpha})n(\boldsymbol{\beta}) - A(\boldsymbol{\alpha},\boldsymbol{\beta})p(\boldsymbol{\alpha}-\boldsymbol{\beta})\right)}{(1+\lambda)^2 - p^2(\boldsymbol{\alpha}-\boldsymbol{\beta})}, \tag{A.29}$$

$$\hat{H}^{\perp}(2,1) = \hat{H}^{\perp}(1,2), \qquad \hat{H}^{\perp}(2,2) = \hat{H}^{\perp}(1,1)\Big|_{(\boldsymbol{\beta},\boldsymbol{\alpha})}. \tag{A.30}$$

Consequently, the eigenvalues of $H^{\perp}$ *are the two (possibly identical) eigenvalues of* $\hat{H}^{\perp}$*, each with multiplicity* $d-1$*.*

Proof. The Hessian has 2×2 block structure. Each block is $d \times d$ and has one value for all diagonal elements and one value for all off-diagonal elements; thus it can be written as $x\mathbf{e}_1\mathbf{e}_1^* + (y-x)I_d$. Applying the procedure in subsection 2.1 yields a transverse Hessian $H^{\perp}$ with 2×2 block structure and each block of the form $(y-x)I_{d-1}$. Consequently, we can write $H^{\perp} = \hat{H}^{\perp} \otimes I_{d-1}$ and the eigenvalues of $H^{\perp}$ are the two (possibly identical) eigenvalues of $\hat{H}^{\perp}$, each with multiplicity $d-1$.

Tracking down the formulas yields

$$\hat{H}^{\perp} = \begin{bmatrix} \frac{\partial^2}{\partial \alpha_j^2} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) - \frac{\partial^2}{\partial \alpha_j \partial \alpha_k} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) & \frac{\partial^2}{\partial \alpha_j \partial \beta_j} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) - \frac{\partial^2}{\partial \alpha_j \partial \beta_k} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \frac{\partial^2}{\partial \alpha_j \partial \beta_j} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) - \frac{\partial^2}{\partial \alpha_j \partial \beta_k} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) & \frac{\partial^2}{\partial \beta_j^2} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) - \frac{\partial^2}{\partial \beta_j \partial \beta_k} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) \end{bmatrix}, \tag{A.31}$$

which is evaluated using (A.18) and (A.19) taking $j \neq k$. Since $(\boldsymbol{\alpha},\boldsymbol{\beta}) = (\alpha\mathbf{1},\beta\mathbf{1})$, we have $n_j(\boldsymbol{\alpha}) = n_k(\boldsymbol{\alpha})$, $n_j(\boldsymbol{\beta}) = n_k(\boldsymbol{\beta})$, $p_j(\boldsymbol{\alpha}-\boldsymbol{\beta}) = p_k(\boldsymbol{\alpha}-\boldsymbol{\beta})$, and $p_{jj}(\boldsymbol{\alpha}-\boldsymbol{\beta}) = -p(\boldsymbol{\alpha}-\boldsymbol{\beta})$, and so we can consolidate somewhat to obtain (A.28) and (A.29). We then obtain (A.30) by symmetry. ∎

Lemma A.5. *The in-plane Hessian with respect to the invariant set* $(\boldsymbol{\alpha},\boldsymbol{\beta}) = (\alpha\mathbf{1},\beta\mathbf{1})$ *is*

$$\hat{H}^{\parallel} = \begin{bmatrix} \frac{\partial^2}{\partial \alpha_j^2} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) + (d-1)\frac{\partial^2}{\partial \alpha_j \partial \alpha_k} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) & \frac{\partial^2}{\partial \alpha_j \partial \beta_j} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) + (d-1)\frac{\partial^2}{\partial \alpha_j \partial \beta_k} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \frac{\partial^2}{\partial \alpha_j \partial \beta_j} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) + (d-1)\frac{\partial^2}{\partial \alpha_j \partial \beta_k} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) & \frac{\partial^2}{\partial \beta_j^2} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) + (d-1)\frac{\partial^2}{\partial \beta_j \partial \beta_k} E_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) \end{bmatrix}. \tag{A.32}$$

Proof. Inspecting the proof of Lemma A.4, the entries deleted to make the transverse Hessian are of the form $(y-x)+dx$, where y was the diagonal entry and x the off-diagonal entry. Thus, similar to (A.31), we obtain (A.32). Since we will not use $\hat{H}^{\parallel}$ in this general form, we will not expand or attempt to simplify further. ∎

A.5.2. Stability transverse to the flow. There are two types of transverse stability to consider.

First, we can consider stability transverse to the invariant set $(\boldsymbol{\alpha},\boldsymbol{\beta}) = (\alpha\mathbf{1},\beta\mathbf{1})$. As developed in subsection 2.1, this stability is determined by the smallest eigenvalue of the transverse Hessian $H^{\perp}$ constructed in Lemma A.4. In Lemma A.4 we showed that $H^{\perp}$ has

the same eigenvalues as the 2×2 matrix $\hat{H}^{\perp}$ and gave formulas for its entries. From there one could write formulas for the eigenvalues, but in the general case the formulas do not aid understanding, and so we suppress them. These eigenvalues are used to determine the stability indicator in subsection 2.1.

Second, we can consider stability transverse to the gradient flow itself. The gradient defines a second transverse Hessian $H^{\perp}_{\nabla}$, whose smallest eigenvalue determines this second stability. Since the gradient has the same symmetry as the invariant set $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\alpha \mathbf{1}, \beta \mathbf{1})$, the eigenvalues of $H^{\perp}$ are also eigenvalues of $H^{\perp}_{\nabla}$. In addition, $H^{\perp}_{\nabla}$ has an eigenvalue obtained by projecting $H^{\parallel}$ orthogonal to ∇E. We can explicitly compute this projection as

(A.33)
$$\frac{1}{\left(\frac{\partial}{\partial \beta_j} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})\right)^2 + \left(\frac{\partial}{\partial \alpha_j} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta})\right)^2} \begin{bmatrix} \frac{\partial}{\partial \beta_j} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta}) & -\frac{\partial}{\partial \alpha_j} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta}) \end{bmatrix} H^{\parallel} \begin{bmatrix} \frac{\partial}{\partial \beta_j} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\ -\frac{\partial}{\partial \alpha_j} E_\lambda(\boldsymbol{\alpha}, \boldsymbol{\beta}) \end{bmatrix}.$$

As with $H^{\parallel}$ itself, the expanded formulas are not enlightening, and so we suppress them. The eigenvalues of $H^{\perp}_{\nabla}$ are used to determine the algorithm progress rate in subsection 2.2, which then determines the expected algorithm time of subsection 2.3.

A.5.3. Behavior near the solution when $\lambda = 0$. When $\lambda = 0$, we know that $E_0(0, \phi) = E_0(\phi, 0) = 0$. Since this is the global minimum, the gradient is zero. In this section we determine the eigenvalues of the Hessian at $(\alpha, \beta) = (0, \phi)$, decomposed into transverse (Lemma A.6) and in-plane (Lemma A.7) portions. We note that although the eigenvalues depend on z, the ratios of eigenvalues depend only on z^2. Thus the algorithm progress rate and estimated time in subsections 2.2 and 2.3 are the same for z and $-z$. We then add the constraint $z = \pm 1$, which allows us to analyze the dependence on ϕ.

Lemma A.6. *If $\lambda = 0$ and $(\alpha, \beta) = (0, \phi)$, then the transverse Hessian $H^{\perp}$ has eigenvalues*

(A.34)
$$\begin{bmatrix} \mu^{\perp} \\ \eta^{\perp} \end{bmatrix} = \frac{1}{1 + 2z\cos^d(\phi) + z^2} \left((1 + z^2) \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \sqrt{(1 - z^2)^2 + 4z^2 \cos^{2d-4}(\phi)} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right),$$

with $0 \le \mu^{\perp} \le \eta^{\perp}$.

Proof. Plugging into Lemma A.4 and simplifying yields

(A.35)
$$\hat{H}^{\perp} = \frac{2}{1 + 2z\cos^d(\phi) + z^2} \begin{bmatrix} 1 & z\cos^{d-2}(\phi) \\ z\cos^{d-2}(\phi) & z^2 \end{bmatrix},$$

from which we can compute the eigenvalues (A.34) directly. ∎

Lemma A.7. *If $\lambda = 0$ and $(\alpha, \beta) = (0, \phi)$, then the in-plane Hessian $H^{\parallel}$ has eigenvalues*

(A.36)
$$\begin{bmatrix} \mu^{\parallel} \\ \eta^{\parallel} \end{bmatrix} = \frac{1 - d\cos^{2d-2}(\phi) + (d-1)\cos^{2d}(\phi)}{(1 + 2z\cos^d(\phi) + z^2)(1 - \cos^{2d}(\phi))}$$
$$\times \left((1 + z^2) \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \sqrt{(1 - z^2)^2 + 4z^2 \left(\frac{\cos^{d-2}(\phi)(1 - d + d\cos^2(\phi) - \cos^{2d}(\phi))}{1 - d\cos^{2d-2}(\phi) + (d-1)\cos^{2d}(\phi)} \right)^2} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right),$$

with $0 \le \mu^{\parallel} \le \eta^{\parallel}$.

Proof. Plugging into Lemma A.5 and simplifying yields

$$H^{\|} = \frac{2(1 - d\cos^{2d-2}(\phi) + (d-1)\cos^{2d}(\phi))}{(1 + 2z\cos^{d}(\phi) + z^2)(1 - \cos^{2d}(\phi))} \times \begin{bmatrix} 1 & z\frac{\cos^{d-2}(\phi)(1-d+d\cos^2(\phi)-\cos^{2d}(\phi))}{1-d\cos^{2d-2}(\phi)+(d-1)\cos^{2d}(\phi)} \\ z\frac{\cos^{d-2}(\phi)(1-d+d\cos^2(\phi)-\cos^{2d}(\phi))}{1-d\cos^{2d-2}(\phi)+(d-1)\cos^{2d}(\phi)} & z^2 \end{bmatrix}, \tag{A.37}$$

from which we can compute the eigenvalues (A.36) directly. ∎

We now add the constraint $z = \pm 1$ in order to perform a more detailed analysis. Lemmas A.8 and A.9 compute the eigenvalues of the transverse and in-plane Hessians and their limits as $\phi \to 0^+$. Lemma A.10 determines the order of the eigenvalues. We observe that as $\phi \to 0^+$, the algorithm progress rate is governed by $\mu^{\|}/\eta^{\perp} \to 0$, and thus convergence to the minimum becomes slower as $\phi \to 0^+$.

Lemma A.8. *If $\lambda = 0$, $(\alpha, \beta) = (0, \phi)$, and $z = \pm 1$, then the transverse Hessian $H^{\perp}$ has eigenvalues*

$$\begin{bmatrix} \mu^{\perp} \\ \eta^{\perp} \end{bmatrix} = \frac{1}{1 + z\cos^{d}(\phi)} \begin{bmatrix} 1 - \cos^{d-2}(\phi) \\ 1 + \cos^{d-2}(\phi) \end{bmatrix}, \tag{A.38}$$

which have limits

$$\lim_{\phi \to 0^+} \begin{bmatrix} \mu^{\perp} \\ \eta^{\perp} \end{bmatrix}_{z=1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \lim_{\phi \to 0^+} \begin{bmatrix} \mu^{\perp} \\ \eta^{\perp} \end{bmatrix}_{z=-1} = \begin{bmatrix} (d-2)/d \\ \infty \end{bmatrix}. \tag{A.39}$$

Proof. Inserting $z^2 = 1$ into Lemma A.6 yields (A.38), from which we can compute the limits. ∎

Lemma A.9. *If $\lambda = 0$, $(\alpha, \beta) = (0, \phi)$, and $z = \pm 1$, then the in-plane Hessian has eigenvalues*

$$\begin{bmatrix} \mu^{\|} \\ \eta^{\|} \end{bmatrix} = \frac{\begin{bmatrix} (1 + \cos^{d}(\phi))(1 - (d-1)\cos^{d-2}(\phi) + (d-1)\cos^{d}(\phi) - \cos^{2d-2}(\phi)) \\ (1 - \cos^{d}(\phi))(1 + (d-1)\cos^{d-2}(\phi) - (d-1)\cos^{d}(\phi) - \cos^{2d-2}(\phi)) \end{bmatrix}}{(1 + z\cos^{d}(\phi))(1 - \cos^{2d}(\phi))}, \tag{A.40}$$

which have limits

$$\lim_{\phi \to 0^+} \begin{bmatrix} \mu^{\|} \\ \eta^{\|} \end{bmatrix}_{z=1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \lim_{\phi \to 0^+} \begin{bmatrix} \mu^{\|} \\ \eta^{\|} \end{bmatrix}_{z=-1} = \begin{bmatrix} 0 \\ 2(d-1)/d \end{bmatrix}. \tag{A.41}$$

Proof. Inserting $z^2 = 1$ into Lemma A.7 yields (A.40), from which we can compute the limits. ∎

Lemma A.10. *The eigenvalues in Lemmas A.8 and A.9 satisfy $0 \le \mu^{\|} \le \mu^{\perp} \le \eta^{\|}$ (and $\mu^{\perp} \le \eta^{\perp}$). For sufficiently small ϕ, we have $\eta^{\|} < \eta^{\perp}$.*

Proof. Algebraic manipulations reduce $\mu^{\|} \leq \mu^{\perp}$ to $d\sin^2(\phi)+2(\cos^d(\phi)-1) \geq 0$. Equality holds at $\phi = 0$, and the derivative of the function on the left is $2d\sin(\phi)\cos(\phi)(1-\cos^{d-2}(\phi))$, which is positive for $\phi > 0$, and so the inequality holds. Similarly, $\mu^{\perp} \leq \eta^{\|}$ reduces to $d\sin^2(\phi) + 2\cos^2(\phi) \geq 0$, which is true. The claim $\eta^{\|} < \eta^{\perp}$ for small ϕ follows from the limits (A.39) and (A.41). ∎

Acknowledgments. I would like to thank Xue Gong, Nathaniel McClatchey, and Todd Young for their suggestions, discussions, and critique regarding this project. I would also like to thank the anonymous referees for their constructive comments.

REFERENCES

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda, *A scalable optimization approach for fitting canonical tensor decompositions*, J. Chemometrics, 25 (2011), pp. 67–86, https://doi.org/10.1002/cem.1335.

[2] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Natl. Acad. Sci. USA, 99 (2002), pp. 10246–10251, https://doi.org/10.1073/pnas.112329799.

[3] G. Beylkin and M. J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159, https://doi.org/10.1137/040604959.

[4] P. Breiding, *Numerical and Statistical Aspects of Tensor Decompositions*, Ph.D. thesis, Technischen Universität Berlin, Berlin, Germany, 2017, http://personal-homepages.mis.mpg.de/breiding/breiding_dissertation.pdf.

[5] P. Breiding and N. Vannieuwenhoven, *The condition number of join decompositions*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 287–309, https://doi.org/10.1137/17M1142880.

[6] P. Breiding and N. Vannieuwenhoven, *Convergence analysis of Riemannian Gauss-Newton methods and its connection with the geometric condition number*, Appl. Math. Lett., 78 (2018), pp. 42–50, https://doi.org/10.1016/j.aml.2017.10.009.

[7] P. Breiding and N. Vannieuwenhoven, *A Riemannian trust region method for the canonical tensor rank approximation problem*, SIAM J. Optim., 28 (2018), pp. 2435–2465, https://doi.org/10.1137/17M114618X.

[8] R. Bro, *Multi-way Analysis in the Food Industry: Models, Algorithms, and Applications*, Ph.D. thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, 1998.

[9] J. Buczyński and J. M. Landsberg, *On the third secant variety*, J. Algebraic Combin., 40 (2014), pp. 475–502, https://doi.org/10.1007/s10801-013-0495-0.

[10] P. Comon, X. Luciani, and A. L. F. de Almeida, *Tensor decompositions, alternating least squares and other tales*, J. Chemometrics, 23 (2009), pp. 393–405, https://doi.org/10.1002/cem.1236.

[11] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127, https://doi.org/10.1137/06066518X.

[12] H. De Sterck, *A nonlinear GMRES optimization algorithm for canonical tensor decomposition*, SIAM J. Sci. Comput., 34 (2012), pp. A1351–A1379, https://doi.org/10.1137/110835530.

[13] H. De Sterck and K. Miller, *An adaptive algebraic multigrid algorithm for low-rank canonical tensor decomposition*, SIAM J. Sci. Comput., 35 (2013), pp. B1–B24, https://doi.org/10.1137/110855934.

[14] H. De Sterck and M. Winlaw, *A nonlinearly preconditioned conjugate gradient algorithm for rank-R canonical tensor approximation*, Numer. Linear Algebra Appl., 22 (2015), pp. 410–432, https://doi.org/10.1002/nla.1963.

[15] M. Espig and W. Hackbusch, *A regularized Newton method for the efficient approximation of tensors represented in the canonical tensor format*, Numer. Math., 122 (2012), pp. 489–525, https://doi.org/10.1007/s00211-012-0465-9.

[16] M. Espig, W. Hackbusch, T. Rohwedder, and R. Schneider, *Variational calculus with sums of elementary tensors of fixed rank*, Numer. Math., 122 (2012), pp. 469–488, https://doi.org/10.1007/s00211-012-0464-x.

[17] X. Gong, M. J. Mohlenkamp, and T. R. Young, *The optimization landscape for fitting a rank-2 tensor with a rank-1 tensor*, SIAM J. Appl. Dyn. Syst., 17 (2018), pp. 1432–1477, https://doi.org/

10.1137/17M112213X.
[18] W. Hackbusch, *Efficient convolution with the Newton potential in d dimensions*, Numer. Math., 110 (2008), pp. 449–489, https://doi.org/10.1007/s00211-008-0171-9.
[19] R. A. Harshman, *Foundations of the PARAFAC Procedure: Model and Conditions for an "Explanatory" Multi-mode Factor Analysis*, Working Papers in Phonetics 16, UCLA, Los Angeles, CA, 1970, http://www.psychology.uwo.ca/faculty/harshman/wpppfac0.pdf.
[20] P. Hopke, P. Paatero, H. Jia, R. Ross, and R. Harshman, *Three-way (PARAFAC) factor analysis: Examination and comparison of alternative computational methods as applied to ill-conditioned data*, Chemometrics Intell. Lab. Syst., 43 (1998), pp. 25–42, https://doi.org/10.1016/S0169-7439(98)00077-X.
[21] A. S. Householder, *Unitary triangularization of a nonsymmetric matrix*, J. Assoc. Comput. Mach., 5 (1958), pp. 339–342, https://doi.org/10.1145/320941.320947.
[22] V. A. Kazeev and E. E. Tyrtyshnikov, *The structure of the Hessian and the efficient implementation of Newton's method in the problem of the canonical approximation of tensors*, Zh. Vychisl. Mat. Mat. Fiz., 50 (2010), pp. 979–998, https://doi.org/10.1134/S0965542510060011.
[23] W. P. Krijnen, T. K. Dijkstra, and A. Stegeman, *On the non-existence of optimal solutions and the occurrence of "degeneracy" in the CANDECOMP/PARAFAC model*, Psychometrika, 73 (2008), pp. 431–439, https://doi.org/10.1007/s11336-008-9056-1.
[24] S. E. Leurgans, R. T. Ross, and R. B. Abel, *A decomposition for three-way arrays*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 1064–1083, https://doi.org/10.1137/0614071.
[25] N. Li, *Variants of ALS on Tensor Decompositions and Applications*, Ph.D. thesis, Clarkson University, Potsdam, NY, 2013.
[26] N. Li, S. Kindermann, and C. Navasca, *Some convergence results on the regularized alternating least-squares method for tensor decomposition*, Linear Algebra Appl., 438 (2013), pp. 796–812, https://doi.org/10.1016/j.laa.2011.12.002.
[27] N. McClatchey, *Tensors: An Adaptive Approximation Algorithm, Convergence in Direction, and Connectedness Properties*, Ph.D. thesis, Ohio University, Athens, OH, 2018, http://rave.ohiolink.edu/etdc/view?acc_num=ohiou1520508234977924.
[28] B. C. Mitchell and D. S. Burdick, *Slowly converging PARAFAC sequences: Swamps and two-factor degeneracies*, J. Chemometrics, 8 (1994), pp. 155–168, https://doi.org/10.1002/cem.1180080207.
[29] M. J. Mohlenkamp, *Musings on multilinear fitting*, Linear Algebra Appl., 438 (2013), pp. 834–852, https://doi.org/10.1016/j.laa.2011.04.019.
[30] M. J. Mohlenkamp and L. Monzón, *Trigonometric identities and sums of separable functions*, Math. Intelligencer, 27 (2005), pp. 65–69, https://doi.org/10.1007/BF02985795.
[31] C. Navasca, L. D. Lathauwer, and S. Kinderman, *Swamp reducing technique for tensor decomposition*, in Proceedings of the 16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, 2008, http://ieeexplore.ieee.org/document/7080724/.
[32] P. Paatero, *A weighted non-negative least squares algorithm for three-way "PARAFAC" factor analysis*, Chemometrics Intell. Lab. Syst., 38 (1997), pp. 223–242, https://doi.org/10.1016/S0169-7439(97)00031-2.
[33] P. Paatero, *The multilinear engine—a table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model*, J. Comput. Graph. Statist., 8 (1999), pp. 854–888, https://doi.org/10.2307/1390831.
[34] P. Paatero, *Construction and analysis of degenerate PARAFAC models*, J. Chemometrics, 14 (2000), pp. 285–299, https://doi.org/10.1002/1099-128X(200005/06)14:3<285::AID-CEM584>3.3.CO;2-T.
[35] A.-H. Phan, P. Tichavský, and A. Cichocki, *Low complexity damped Gauss–Newton algorithms for CANDECOMP/PARAFAC*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 126–147, https://doi.org/10.1137/100808034.
[36] M. Rajih, P. Comon, and R. A. Harshman, *Enhanced line search: A novel method to accelerate PARAFAC*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1128–1147, https://doi.org/10.1137/06065577.
[37] W. S. Rayens and M. C. Mitchell, *Two-factor degeneracies and a stabilization of PARAFAC*, Chemometrics Intell. Lab. Syst., 38 (1997), pp. 173–181.
[38] M. Razaviyayn, M. Hong, and Z.-Q. Luo, *A unified convergence analysis of block successive min-*

imization methods for nonsmooth optimization, SIAM J. Optim., 23 (2013), pp. 1126–1153, https://doi.org/10.1137/120891009.

[39] L. Sorber, I. Domanov, M. Van Barel, and L. De Lathauwer, *Exact line and plane search for tensor optimization*, Comput. Optim. Appl., 63 (2016), pp. 121–142, https://doi.org/10.1007/s10589-015-9761-5.

[40] L. Sorber, M. Van Barel, and L. De Lathauwer, *Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization*, SIAM J. Optim., 23 (2013), pp. 695–720, https://doi.org/10.1137/120868323.

[41] A. Stegeman, *Degeneracy in CANDECOMP/PARAFAC explained for $p \times p \times 2$ arrays of rank $p+1$ or higher*, Psychometrika, 71 (2006), pp. 483–501, https://doi.org/10.1007/s11336-004-1266-6.

[42] A. Stegeman, *Degeneracy in CANDECOMP/PARAFAC and INDSCAL explained for several three-sliced arrays with a two-valued typical rank*, Psychometrika, 72 (2007), pp. 601–619, https://doi.org/10.1007/s11336-007-9022-3.

[43] A. Stegeman, *Low-rank approximation of generic $p \times q \times 2$ arrays and diverging components in the CANDECOMP/PARAFAC model*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 988–1007, https://doi.org/10.1137/050644677.

[44] A. Stegeman, *CANDECOMP/PARAFAC: From diverging components to a decomposition in block terms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 291–316, https://doi.org/10.1137/110825327.

[45] A. Stegeman, *A three-way Jordan canonical form as limit of low-rank tensor approximations*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 624–650, https://doi.org/10.1137/120875806.

[46] A. Stegeman, *Finding the limit of diverging components in three-way CANDECOMP/PARAFAC—a demonstration of its practical merits*, Comput. Statist. Data Anal., 75 (2014), pp. 203–216, https://doi.org/10.1016/j.csda.2014.02.010.

[47] A. Stegeman and L. De Lathauwer, *A method to avoid diverging components in the CANDECOMP/PARAFAC model for generic $I \times J \times 2$ arrays*, SIAM J. Matrix Anal. Appl., 30 (2009), pp. 1614–1638, https://doi.org/10.1137/070692121.

[48] G. Tomasi and R. Bro, *PARAFAC and missing values*, Chemometrics Intell. Lab. Syst., 75 (2005), pp. 163–180, https://doi.org/10.1016/j.chemolab.2004.07.003.

[49] G. Tomasi and R. Bro, *A comparison of algorithms for fitting the PARAFAC model*, Comput. Statist. Data Anal., 50 (2006), pp. 1700–1734, https://doi.org/10.1016/j.csda.2004.11.013.

[50] A. Uschmajew, *Well-posedness of convex maximization problems on Stiefel manifolds and orthogonal tensor product approximations*, Numer. Math., 115 (2010), pp. 309–331, https://doi.org/10.1007/s00211-009-0276-9.