`fieldsMAGMA': A MAGMA-accelerated extension to the `fields' spatial statistics R package

John Paige
Doug Nychka
Dorit Hammerling

NCAR Technical Notes
NCAR/TN-520+STR

National Center for
Atmospheric Research
P. O. Box 3000
Boulder, Colorado
80307-3000
www.ucar.edu

NCAR TECHNICAL NOTES

http://library.ucar.edu/research/publish-technote

The Technical Notes series provides an outlet for a variety of NCAR Manuscripts that contribute in specialized ways to the body of scientific knowledge but that are not yet at a point of a formal journal, monograph or book publication. Reports in this series are issued by the NCAR scientific divisions, serviced by OpenSky and operated through the NCAR Library. Designation symbols for the series include:

EDD – Engineering, Design, or Development Reports
Equipment descriptions, test results, instrumentation, and operating and maintenance manuals.

IA – Instructional Aids
Instruction manuals, bibliographies, film supplements, and other research or instructional aids.

PPR – Program Progress Reports
Field program reports, interim and working reports, survey reports, and plans for experiments.

PROC – Proceedings
Documentation or symposia, colloquia, conferences, workshops, and lectures. (Distribution maybe limited to attendees).

STR – Scientific and Technical Reports
Data compilations, theoretical and numerical investigations, and experimental results.

The National Center for Atmospheric Research (NCAR) is operated by the nonprofit University Corporation for Atmospheric Research (UCAR) under the sponsorship of the National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

National Center for Atmospheric Research
P. O. Box 3000
Boulder, Colorado 80307-3000

NCAR/TN-520+STR
NCAR Technical Note

2015-09

`fieldsMAGMA': A MAGMA-accelerated extension to the `fields' spatial statistics R package

John Paige
Institute for Mathematics and Applied Geosciences,
National Center for Atmospheric Research, Boulder, CO
Douglas Nychka
Institute for Mathematics and Applied Geosciences,
National Center for Atmospheric Research, Boulder, CO
Dorit Hammerling
Institute for Mathematics and Applied Geosciences,
National Center for Atmospheric Research, Boulder, CO

Computational and Information Systems Laboratory (CISL)
Institute for Mathematics Applied to the Geosciences (IMAGe)

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
P. O. Box 3000
BOULDER, COLORADO 80307-3000
ISSN Print Edition 2153-2397
ISSN Electronic Edition 2153-2400

'fieldsMAGMA': A MAGMA-accelerated extension to the 'fields' spatial statistics R package

John Paige, Douglas Nychka, Dorit Hammerling *

September 23, 2015

Abstract

This report introduces the 'fieldsMAGMA' R package, an extension to the 'fields' package for spatial data analysis that is available on github. fieldsMAGMA uses the Cholesky decomposition functionality of the MAGMA multi-GPU, multi-CPU computing library and eliminates some unnecessary distance and covariance calculations to create accelerated versions of spatial statistics methods in fields. We demonstrate the performance of fieldsMAGMA's accelerated functions when applied to simulated datasets and the CO2 dataset available in fields. We show that using the single precision Cholesky decomposition in particular has the potential for vast improvements in the Cholesky decomposition and in spatial likelihood computation time, yet the accuracy of the likelihood maximization is not significantly reduced. We gather some of our timing results on a 2014 MacBook Pro with a stock graphics processing unit (GPU), an NVIDIA GeForce GT 750M with 2048 MB GDDR5 RAM.

Keywords: Spatial statistics, Kriging, High-Performance Computing, GPU, MAGMA

*John Paige, is Research Associate, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley CA 94720 (paigejo@uw.edu), Douglas Nychka, is Senior Scientist, National Center for Atmospheric Research, PO Box 3000, Boulder CO 30307-3000 (nychka@ucar.edu), Dorit Hammerling is Project Scientist II, National Center for Atmospheric Research, PO Box 3000, Boulder CO 30307-3000 (dorith@ucar.edu).

Contents

List of Figures

1 Introduction

Graphics processing units (GPUs) have rapidly evolved from computer chips meant exclusively for controlling graphics displays to today's general purpose GPUs (GPGPUs). GPGPUs are used in many applications, especially those that are highly parallelizeable, and have in some settings enabled substantial speedups over traditional methods of computation, those using only central processing units (CPUs). A single GPU can even be faster than multiple CPUs for certain applications. This is because GPUs have hundreds of computing threads whereas most CPUs have on the order of tens of threads or fewer (Nickolls and Dally 2010, Du et al. 2012).

Yet in spite of GPUs' many advantages over CPUs, CPUs continue to outperform GPUs for some problems. This is why heterogeneous computer architectures—architectures that use CPUs and GPUs together—can perform so well. Algorithms taking full advantage of these hybrid architectures use CPUs and GPUs each for the portions of computation they are most suited for, and are typically faster than algorithms using only CPUs or only GPUs (Nickolls and Dally 2010). Since heterogeneous architectures are common among supercomputers and ubiquitous among personal computers, it is important for algorithms to achieve a high level of performance on these systems.

The '`fields`' spatial statistics package (freely available on CRAN) is an `R` package that can currently only run its computations with CPUs. It is widely used, but has computational limits when estimating spatial processes using several tens of thousands of observations or more. This is because it relies on the 'Kriging' algorithm, which is central to spatial statistics and is $\mathcal{O}(n^3)$ in computation time for n spatial observations (Katzfuss and Cressie 2012). Since many spatial datasets contain millions or even tens of millions of observations, this represents a serious problem (Katzfuss and Cressie 2012). The biggest computational hurdle in Kriging is in computing the data's likelihood given a spatial model and its parameters, and the most computationally intensive step in the likelihood calculation is the Cholesky decomposition of the data covariance matrix. Although there are many ways to get around this problem using approximations, such as inducing sparsity into the covariance matrix, or using reduced rank approximations (*e.g.* Nychka et al. 2014b, Nychka et al. 2014a, Cressie and Johannesson 2008), these methods are approximations and have disadvantages in certain circumstances (Stein 2014).

Recently, Paige et al. (2015) introduced an extension to the `fields` package that incor-

porated the freely available Matrix Algebra on GPU and Multicore Architectures (MAGMA) library. However, the code introduced in Paige et al. (2015) was not yet a pre-bundled package format, which made it more cumbersome to load into the `R` environment. This report introduces the `fieldsMAGMA R` package, which adds single-precision support to the accelerated functions from Paige et al. (2015) and bundles the modified functions into a package that is publicly available on github. In addition, certain unnecessary covariance and distance computations currently used in `fields`' Maximum Likelihood Estimation (MLE) algorithm are removed to further accelerate the MLE process.

The MAGMA library (Tomov et al. 2009) exploits the power of heterogeneous computer systems to accelerate matrix algebra routines such as the Cholesky decomposition. In Paige et al. (2015) the Cholesky decomposition speedup using one GPU for 10,000 spatial observations (generating a 10,000 $\times$ 10,000 covariance matrix) was 32.3 times when on Caldera comptuational nodes on the Yellowstone supercomputer. The resulting speedup for the example spatial analysis workflow was 2.4 due to the unaccelerated portions of the workflow. These speedups did not take advantage of the performance benefits in GPUs when using single rather than double precision, however, and this report illustrates the importance of single-precision calculations.

Although some of the most advanced NVIDIA GPUs achieve a two times speedup when using single rather than double precision floating point operations, previous GPU architectures have eight times speedup or more (Du et al. 2012). Even a computer as new as the mid-2014 MacBook Pro used in this report with its NVIDIA GeForce GT 750M GPU has an 8:1 single to double precision performance advantage (Brodtkorb et al. 2013, Du et al. 2012). In addition, since single precision values take half the space of double precision values, they can be transmitted between CPUs and GPUs twice as fast and matrices with twice the number of elements can be stored on GPU memory at once. Fortunately, MAGMA has support for both single and double precision Cholesky decompositions so that even users without the most advanced GPUs are able to see considerable speedup when performing the Cholesky decomposition.

This technical report will use timings from a mid-2014, 15" MacBook Pro with a NVIDIA GeForce GT 750M GPU with 2048 MB GDDR5 memory, clocked up to 5.0 GHz, and 384 cores clocked at approximately 967 MHz. It also is fitted with a 4-core, 2.8 GHz Intel Core i7 CPU with 16 GB memory. It should be emphasized that the GPU was the most basic GPU

available with the computer when it was purchased, shipped by default with the computer at the time. Instructions for installing MAGMA and `fieldsMAGMA` are given in this report that extend those given in Paige et al. (2015).

The rest of this technical report is organized as follows. Section 2 provides timing and speedup results for accelerated and unaccelerated implementations of the Cholesky decomposition, `fields`' `mKrig` function, and a simple spatial analysis workflow. Section 3 compares data likelihood calculations as determined by single and double precision accelerated `mKrig`. Section 4 concludes by interpreting the timing and single versus double precision accuracy results. It then goes on to discuss possible future directions for this line of research. Instructions for installing MAGMA on the MacBook Pro are given in the appendix along with supplemental results illustrating the accuracy of the accelerated single precision likelihood calculations.

2 Timing Results

In this section we give timings and speedups for unaccelerated and accelerated versions of the Cholesky decomposition, `mKrig`, and a sample spatial analysis workflow. More specifically, results are plotted for the default `R` implementation, and MAGMA-accelerated versions in single and double precision with one GPU. For the accelerated implementations, we plot results where the factored matrix is copied before being decomposed and also where the Cholesky decomposition is computed in place, overwriting the original factored matrix. Note that the x axis of each plot is the number of spatial observations. Given n observations, the resulting factored covariance matrix is $n \times n$ and contains n^2 elements.

Figure 1 shows times and speedups for the Cholesky decomposition (computation times are plotted on a log scale). For 10,000 observations, the default Cholesky decomposition took 2 minutes and 7 seconds. In comparison, the double precision accelerated Cholesky decomposition took 13.3 seconds when copying the factored matrix and 12.8 seconds when performing calculations in place, achieving 9.6 and 9.9 times speedup respectively. The single precision accelerated decomposition only took 2.1 (1.7) seconds for 10,000 observations when copying the factored matrix (performing the calculations in place), achieving 60.5 (76.0) times speedup. Although the double precision accelerated Cholesky decomposition appears to peak at approximately 10 times speedup due the GPU reaching near-maximum efficiency, the single precision accelerated decomposition has not peaked after 12,500 observations (meaning

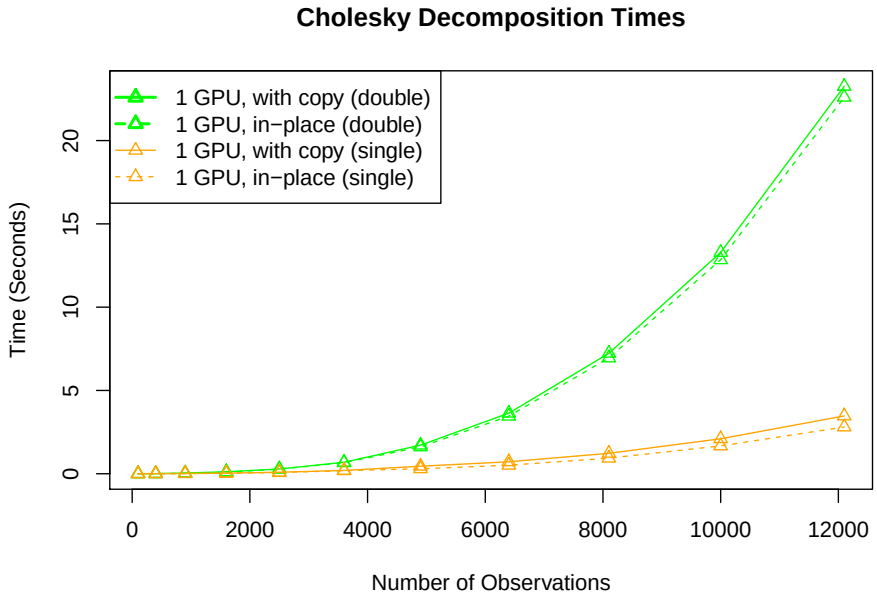
a 12,500 $\times$ 12,500 sized matrix).

Timings for the unaccelerated and accelerated versions of `fields`' `mKrig` function are shown in Figure 2. All calculations for `mKrig` are performed using subsets of the `CO2` dataset in `fields`, where the data subsets are determined using a latitude/longitude window centered at zero latitude and zero longitude. For just over 10,000 observations, `fields`' `mKrig` function takes 2 minutes 25 seconds, the accelerated function takes 20.2 seconds with one GPU in double precision and 8.1 seconds in single precision. Hence, the double precision speedup is 7.2 times and the single precision speedup is 17.9 times. Both single and double precision speedups increase as matrix sizes increase throughout the range of sizes used in our testing (up to 10,000 observations).

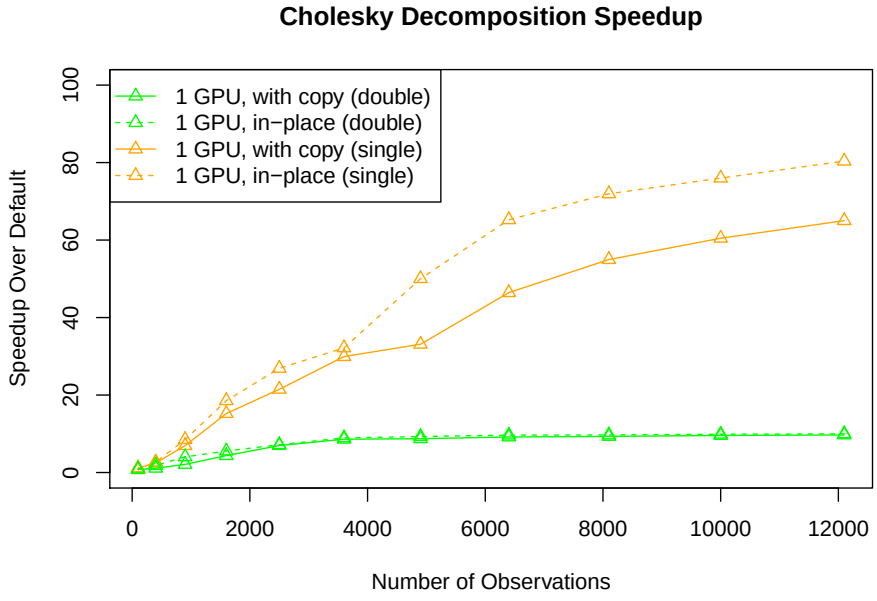
Figure 3 displays the timings and speedups of a spatial problem workflow. This is the same workflow as described in Paige et al. (2015), and it uses the same subsets of the `CO2` dataset in `fields` as the `mKrig` timings (described above). The spatial analysis workflow involves computing a Maximum likelihood Estimate (MLE) of λ, the Kriging smoothing parameter, for a fixed θ, the spatial scale parameter of the exponential covariance model, and then performing Kriging for the MLE of λ and the fixed θ. In the assumed exponential covariance model, the correlation between two observations is:

$$k(x_i, x_j) = \begin{cases} e^{-|x_i - x_j|/\theta}, & i \neq j \\ e^{-|x_i - x_j|/\theta} + \lambda, & i = j \end{cases},$$

for locations x_i and x_j. After performing the MLE process, we calculate a prediction surface on an 80 $\times$ 80 grid. Finally, we estimate the surface's theoretical standard error as a function of space by taking five random draws from the conditional distribution of the spatial process given the observations using the `sim.mKrig.approx` method in `fields`. Overall, 11 Cholesky decompositions are required. For approximately 10,000 observations, the default (unaccelerated) workflow took 26 minutes 55 seconds. The accelerated workflow took 3 minutes 44 seconds and 1 minute 38 seconds for double and single precision respectively which corresponds to 7.2 and 16.4 times speedup respectively. Just as for the `mKrig` timings, speedup continues to increase as the number of observations increases throughout the range of number of observations tested (up to 10,000 observations). The double precision speedup appears to be nearing its theoretical asymptotic supremum of approximately 9.8 times speedup—the maximum of the Cholesky decomposition—as the decomposition increasingly dominates

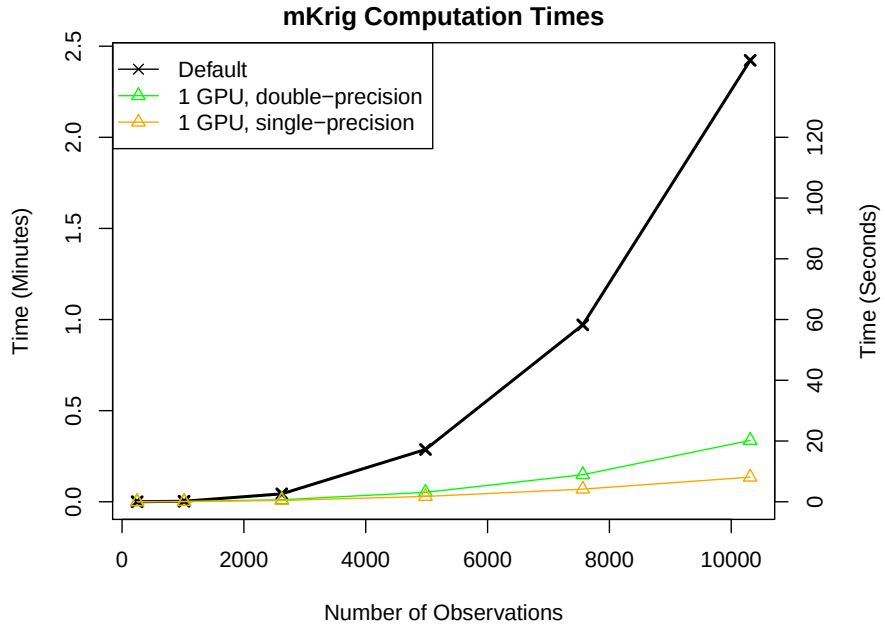


(a)

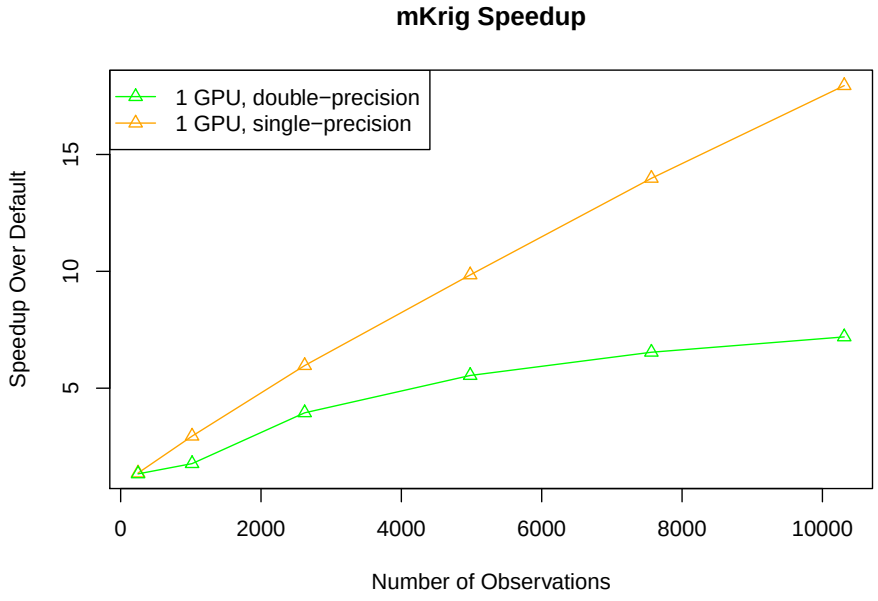


(b)

Figure 1: Computation time (a) and speedup factor (b) of the MAGMA-accelerated implementations of the Cholesky decomposition versus the default implementation (b) are plotted. Times for in-place calculations without setting the lower triangular portion of the matrix to zero are shown as dotted lines, while non-in-place (listed as "with copy") calculation times when setting the lower triangular portion of the decomposition to zero are shown as solid lines. Note that the computation time shown in (a) is plotted on a log scale.



(a)



(b)

Figure 2: Computation time of the `mKrig` function in `fields` for default (black) and accelerated implementations (green and orange) (a) and speedup factor versus the default implementation (b) are plotted. All calculations for accelerated Cholesky decompositions in `fields` are performed in-place while zeroing out the lower triangular portion of the matrix.

the workflow computation. The single precision workflow speedup of 16.4 is still far from its theoretical asymptote of over 80 times speedup, and is increasing with the number of observations at a much faster rate than the double precision workflow speedup.

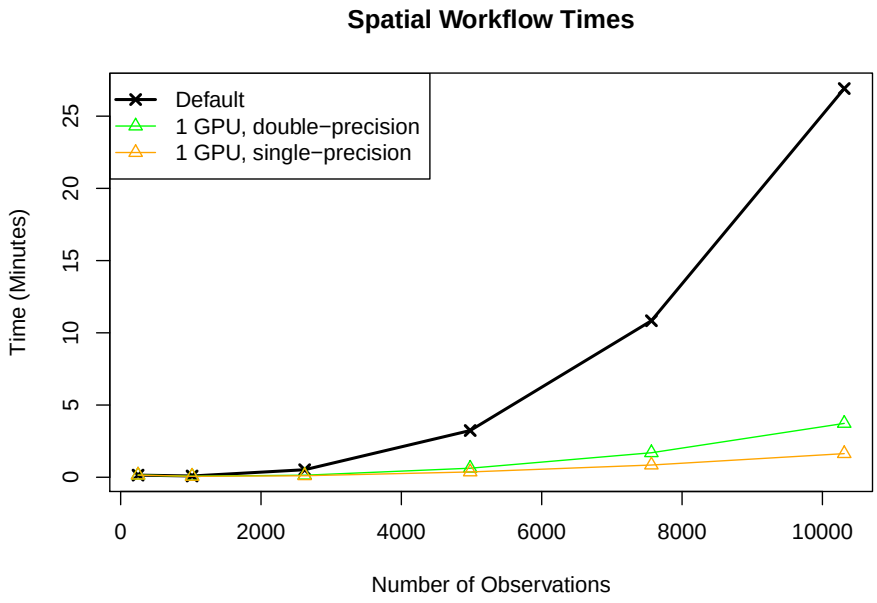
3 Single Precision Likelihood Calculation Accuracy

While double precision floating point numbers use 64 bits to represent a number, single precision floating point numbers only use 32. Because of this, sacrifices must be made in order to represent a number by rounding off the least significant digits. In this section we test the accuracy of the data log likelihood calculated in the spatial analysis using a single precision Cholesky decomposition as compared to that calculated with double precision.

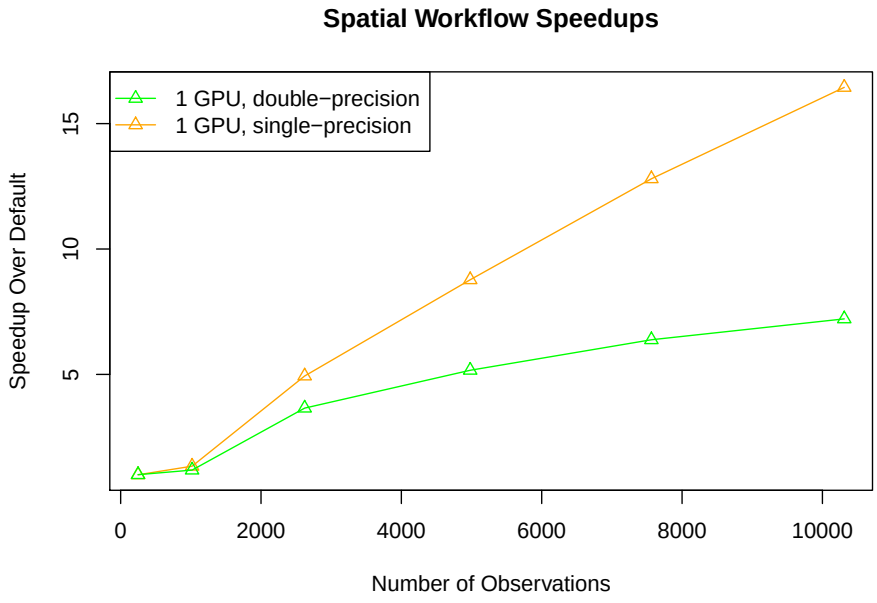
As an initial test, we generated observations on a $[0, 1] \times [0, 1]$ square using an exponential covariance model with different sets of parameters. We varied λ, the Kriging smoothing parameter, as well as θ, the exponential scale parameter. We then calculated the log likelihood of the observations using single and double precision Cholesky decompositions for 5,000, 10,000, 15,000, and 20,000 observations. For each set of true parameters, we generate observations multiple times: 12 times for 5,000 observations, 6 times for 10,000 observations, 4 times for 15,000 observations, and 3 times for 20,000 observations. Plotted in Figures 4-7 are interpolated maximum and average absolute differences between single and double precision log-likelihoods. We then use thin plate splines generated by the `Tps` function in `fields` to interpolate the log-likelihood samples (shown as black dots in the Figures).

As shown in Figures 4-7, absolute difference between single and double precision log-likelihoods for true parameters tends to increase as λ increases over 1. Also, differences increase when λ decreases much below 0.01, when the number of observations increase, and when θ rises above 1. However, in no case did log-likelihood differences increase far over 2 and average log-likelihood differences did not increase far above 1.5. In these cases, the true parameters were fairly extreme compared to the size of the spatial domain, a 1×1 unit square. Note that the 95 percent confidence interval for the true parameters can be calculated by subtracting the 95th percentile of the $\chi^2_2/2$ distribution, which is approximately 3. Still, even if the true parameter log-likelihood has a significant difference, it may not affect the accuracy of the confidence set if the log-likelihood difference between the true parameters and the MLE parameters is essentially unchanged.

In order to test the accuracy of confidence sets created with single precision Cholesky

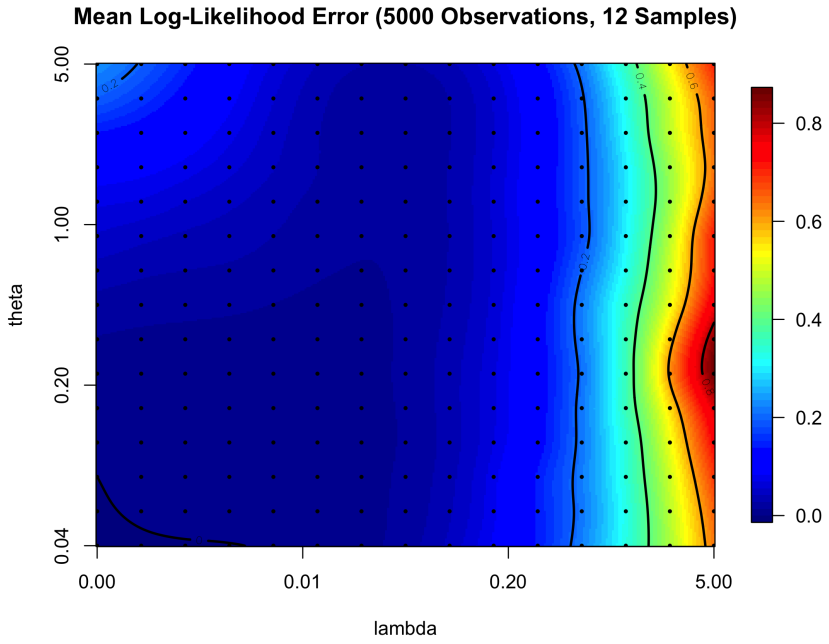


(a)

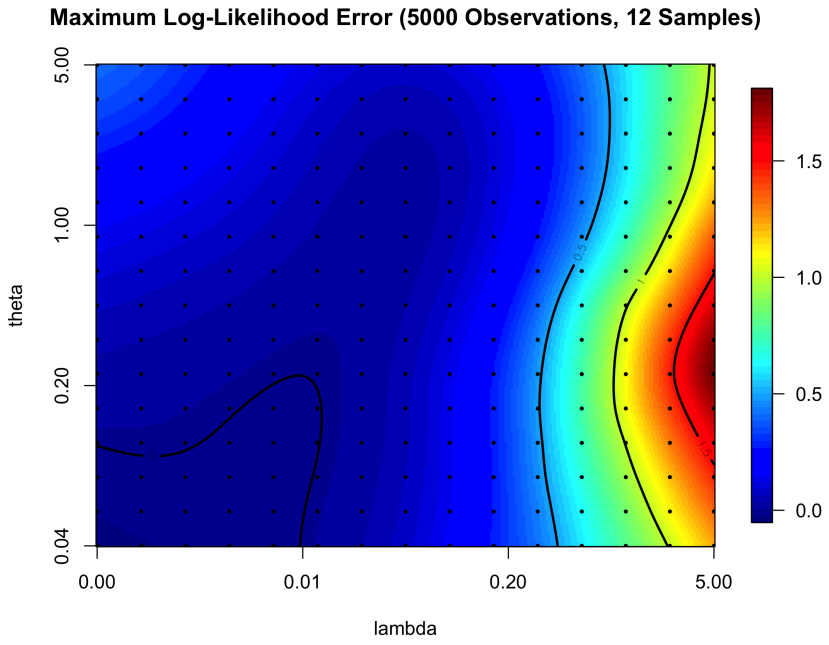


(b)

Figure 3: Computation time for spatial analysis of the `CO2` dataset in `fields` for default (black) and accelerated workflows (green and orange) (a) and speedup factor over the default implementation (b). Spatial analysis includes 11 parameter samples for an exponential covariance function each including a Cholesky decomposition, generating a prediction of the spatial process, and estimating predictive error.

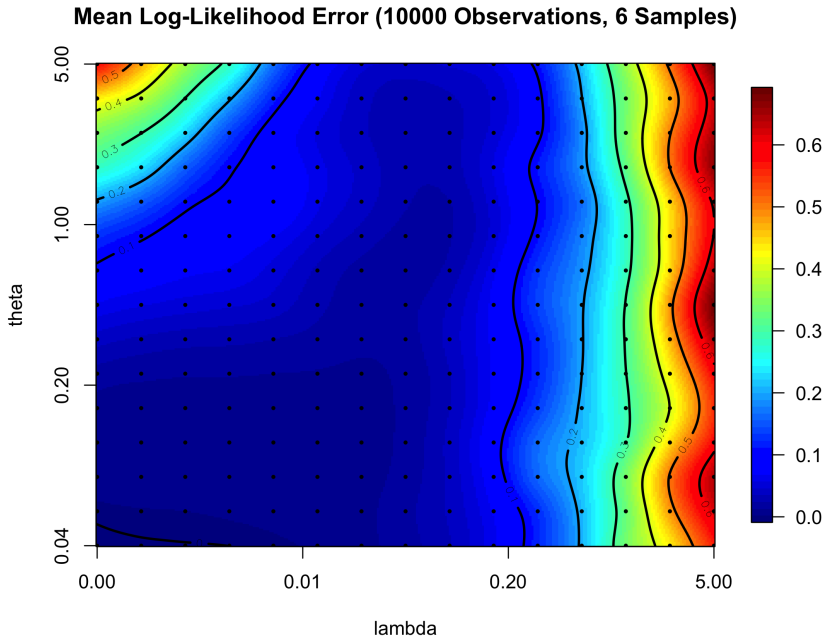


(a)

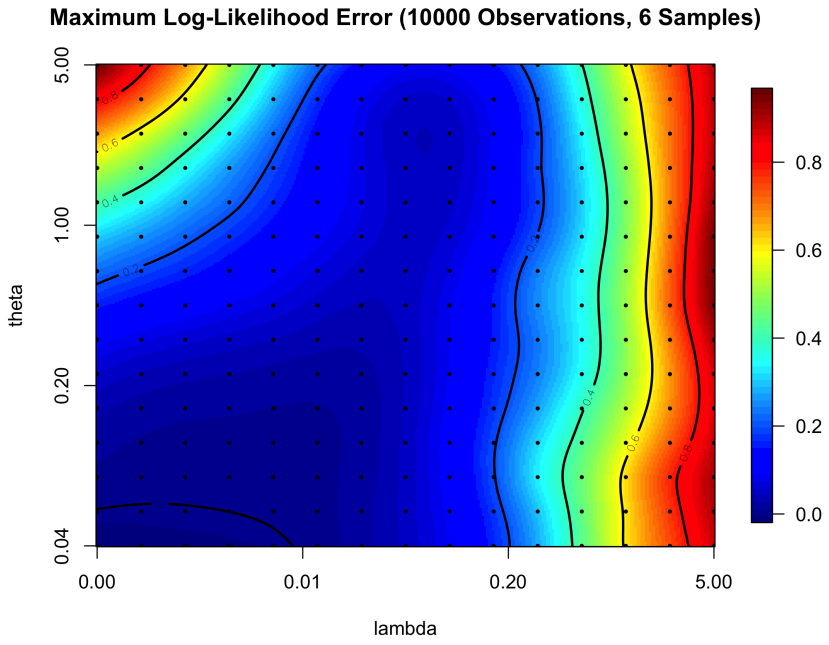


(b)

Figure 4: Mean (a) and max (b) "error" (absolute difference) of the true model parameter log-likelihood across multiple simulated datasets as calculated using single versus double precision Cholesky decompositions. The sampled log-likelihoods are shown as black dots, and the plotted surface is generated using the `Tps` function from `fields`, which uses thin plate splines to generate an interpolating surface.

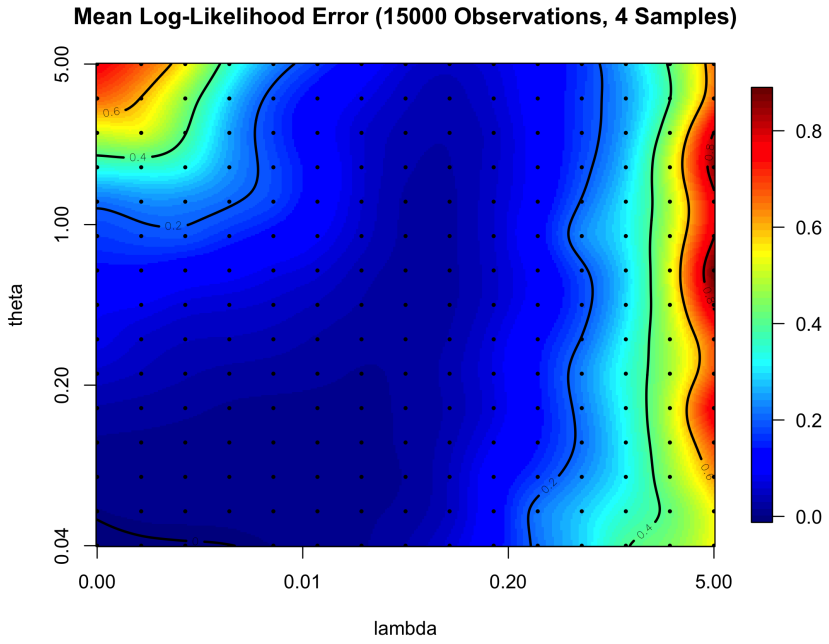


(a)

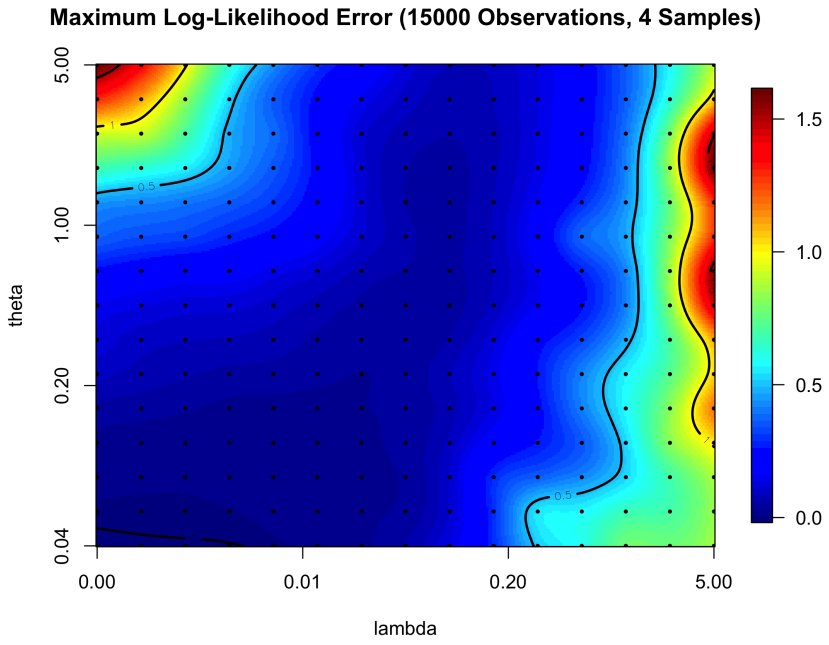


(b)

Figure 5: Mean (a) and max (b) "error" (absolute difference) of the true model parameter log-likelihood across multiple simulated datasets as calculated using single versus double precision Cholesky decompositions. The sampled log-likelihoods are shown as black dots, and the plotted surface is generated using the `Tps` function from `fields`, which uses thin plate splines to generate an interpolating surface.

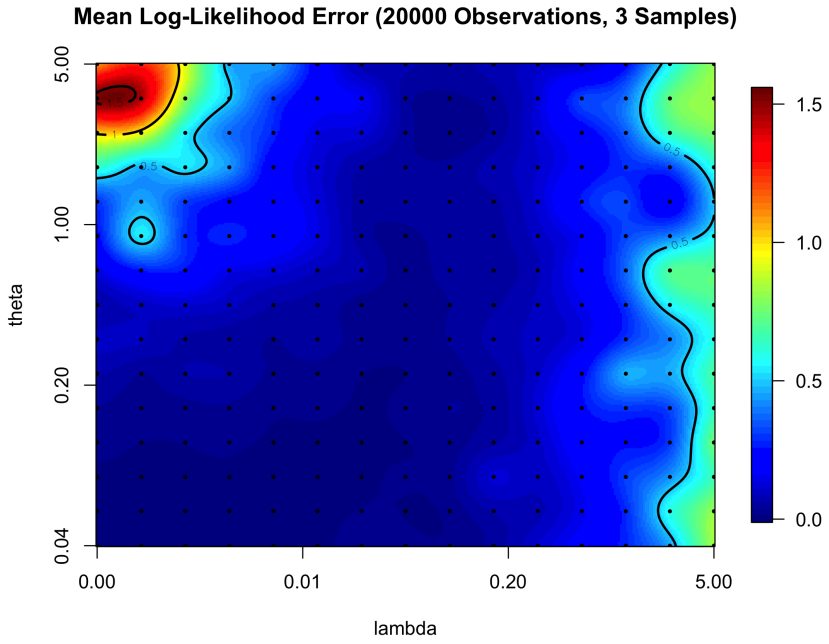


(a)

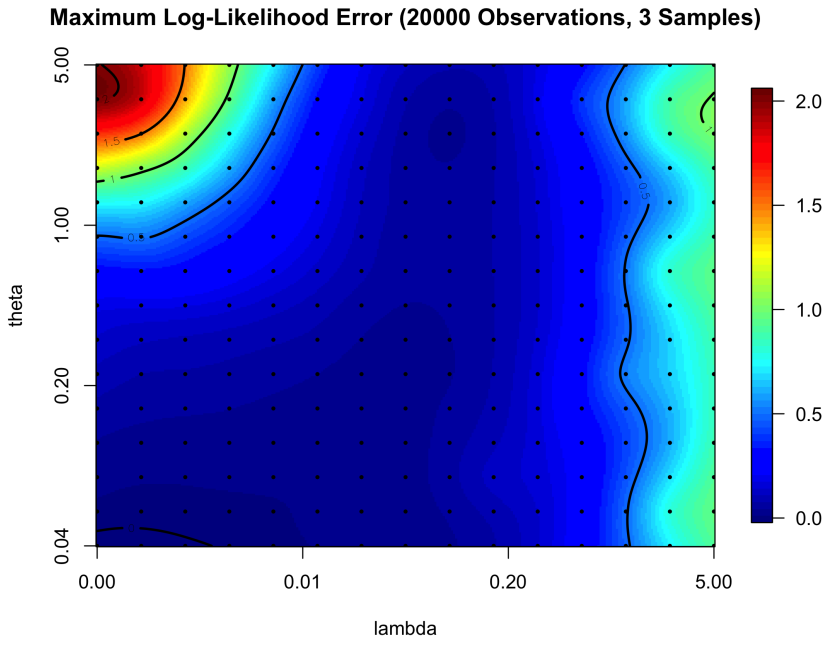


(b)

Figure 6: Mean (a) and max (b) "error" (absolute difference) of the true model parameter log-likelihood across multiple simulated datasets as calculated using single versus double precision Cholesky decompositions. The sampled log-likelihoods are shown as black dots, and the plotted surface is generated using the `Tps` function from `fields`, which uses thin plate splines to generate an interpolating surface.



(a)



(b)

Figure 7: Mean (a) and max (b) "error" (absolute difference) of the true model parameter log-likelihood across multiple simulated datasets as calculated using single versus double precision Cholesky decompositions. The sampled log-likelihoods are shown as black dots, and the plotted surface is generated using the `Tps` function from `fields`, which uses thin plate splines to generate an interpolating surface.

decompositions, we begin by fixing a set of 'true' spatial parameters for an exponential covariance Gaussian Process. We then simulate the spatial field by generating observations on a $[0,1] \times [0,1]$ square, and calculate the log-likelihood of the generated data for a grid of 'guessed' parameter values for an exponential covariance model using single and double precision Cholesky factorizations. We compare these log-likelihood grids for several sets of true spatial parameters. We choose five sets of true parameters that we feel best represent the extreme and less extreme scenarios given the $[0,1] \times [0,1]$ spatial domain of the generated data. We choose the following sets of true parameters:

θ	λ
0.05	0.05
5	0.05
0.2	0.1
0.05	1
5	1

Since the domain was only of width 1 (or $\sqrt{2}$ across the diagonal of the square), we decided to only increase θ up to 5, since this is already 5 times the size of the spatial domain.

We performed all calculations for three different numbers of observations: 1,000, 5,000, and 10,000. In these cases we generated observations and recalculated the log-likelihood grid 10, 2, and 1 times. We report the single sample with the biggest difference between single and double precision MLE log-likelihoods. The log-likelihood surface was sampled on a 25×25 parameter grid spaced on a logarithmic scale from 0.01 to 5. The resolution was low because of the computational difficulty of calculating the log-likelihood, but we approximated a higher resolution log-likelihood using the `Tps` method in fields. `Tps` uses thin-plate splines to approximate multidimensional surfaces. Along with approximated log-likelihood surfaces, we plot the approximated 95 percent confidence sets of the true parameters for both double and single precision. We show each on their corresponding double and single precision log-likelihood surfaces, and both sets are plotted together on another double precision log-likelihood surface for the same dataset. In addition, we show a thin-plate spline approximation for the absolute difference between the single and double precision log-likelihood surfaces and plot the confidence sets on that surface as well for reference.

Over these surfaces, we also plot the true parameter values and the maximum likelihood estimates (MLEs) calculated independently using single and double precision maximum likeli-

hood estimation. When multiple sets of observations are generated (as in the 1,000 and 5,000 observation scenarios), the surfaces, confidence sets, and parameters for generated datasets causing the largest difference between single and double precision MLE log-likelihoods are shown.

Results are plotted in Figures 8-12 for 1,000 observations and in the appendix for 5,000 and 10,000 observations. In all cases the MLE values estimated with single and double precision log-likelihood calculations were similar in log-likelihood, and the single and double precision MLE parameters were also similar in terms of their actual values. At times the single precision MLE was actually better than the double precision MLE, and, of course, the opposite was also sometimes true. We believe this is due to chance and inconsistencies in the `optim` function used to determine the MLE of λ rather than systematic error. The true parameters appeared on the same side of the contours of the single and double precision 95 percent confidence sets in all but ones case (1000 observations, $\theta = 5$, $lambda = 1$). In that instance, neither the double nor single precision confidence sets contained the true parameters, and the parameters were fairly extreme for data on a unit square. In most of the Figures the single and double precision MLEs are plotted almost on top of each other.

In a separate test, we generated 10,000 observation locations in a grid on a $[0, 1] \times [0, 1]$ square, simulating observation values assuming an exponential covariance model and varying the exponential scale, θ and the Kriging smoothing parameter, λ. In addition, we moved a single observation closer and closer to one of its closest neighbors until the single precision Cholesky decomposition resulted in an error due to the covariance matrix being numerically singular. Varying θ from .01 to 10^{17} and λ from 10^{-5} to 1, we found that in most cases `mKrig` returns an error before the matrix becomes singular. In fact, `mKrig` returns an error whenever any observations have equal coordinates when rounded to eight decimal places. However, if λ was 10^{-3} or smaller, the closest observations were $\frac{1}{99} \cdot 10^{-6}$ units apart or less, and θ was larger than 10,000, the the single precision Cholesky often failed when the double precision Cholesky did not necessarily.

4 Discussion and Conclusions

Just as in Paige et al. (2015), speedups tend to increase as the number of observations and therefore the covariance matrix dimensions increase for accelerated `mKrig` and the spatial workflow, although the Cholesky decomposition speedup peaks at 10 times speedup. How-

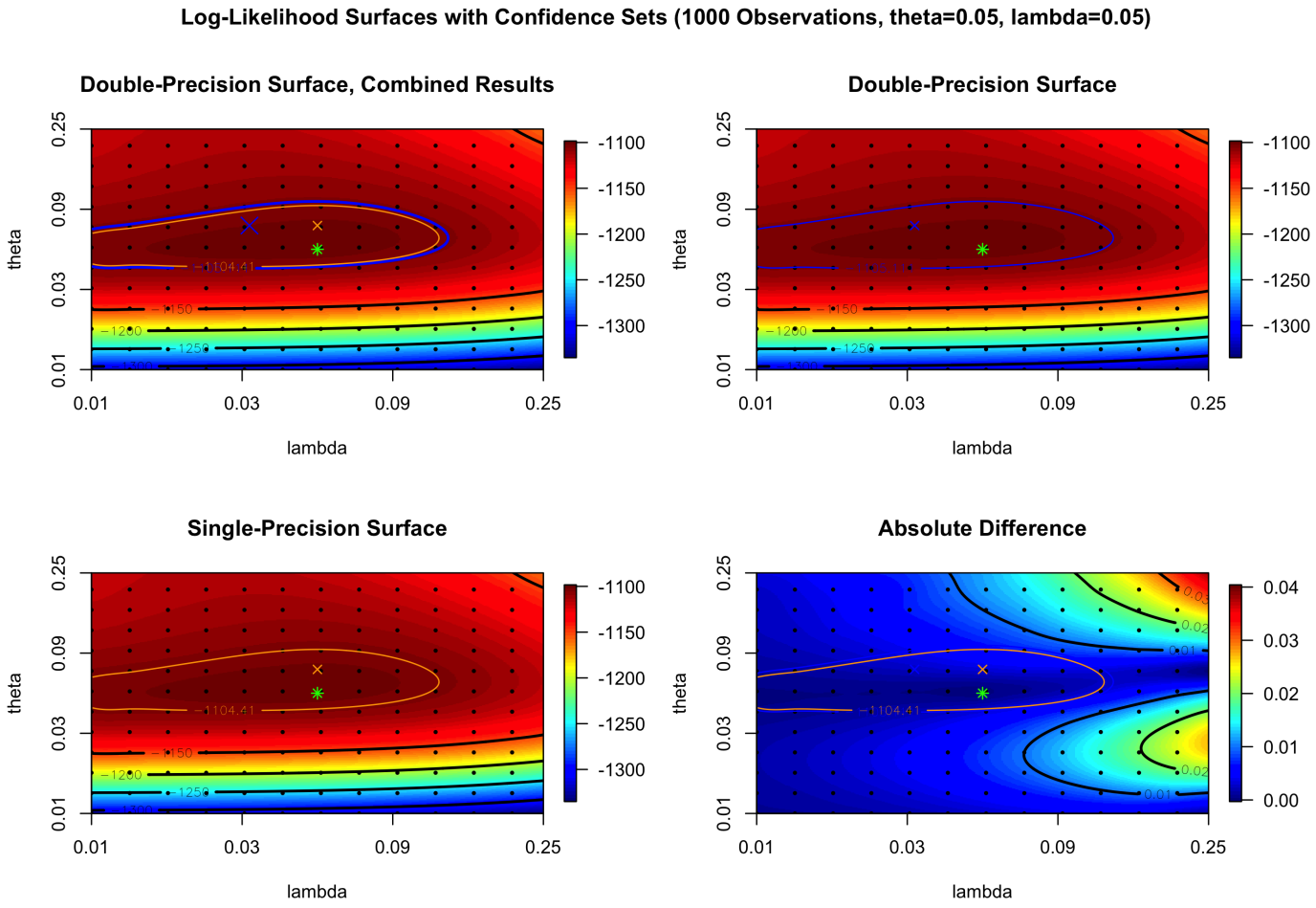
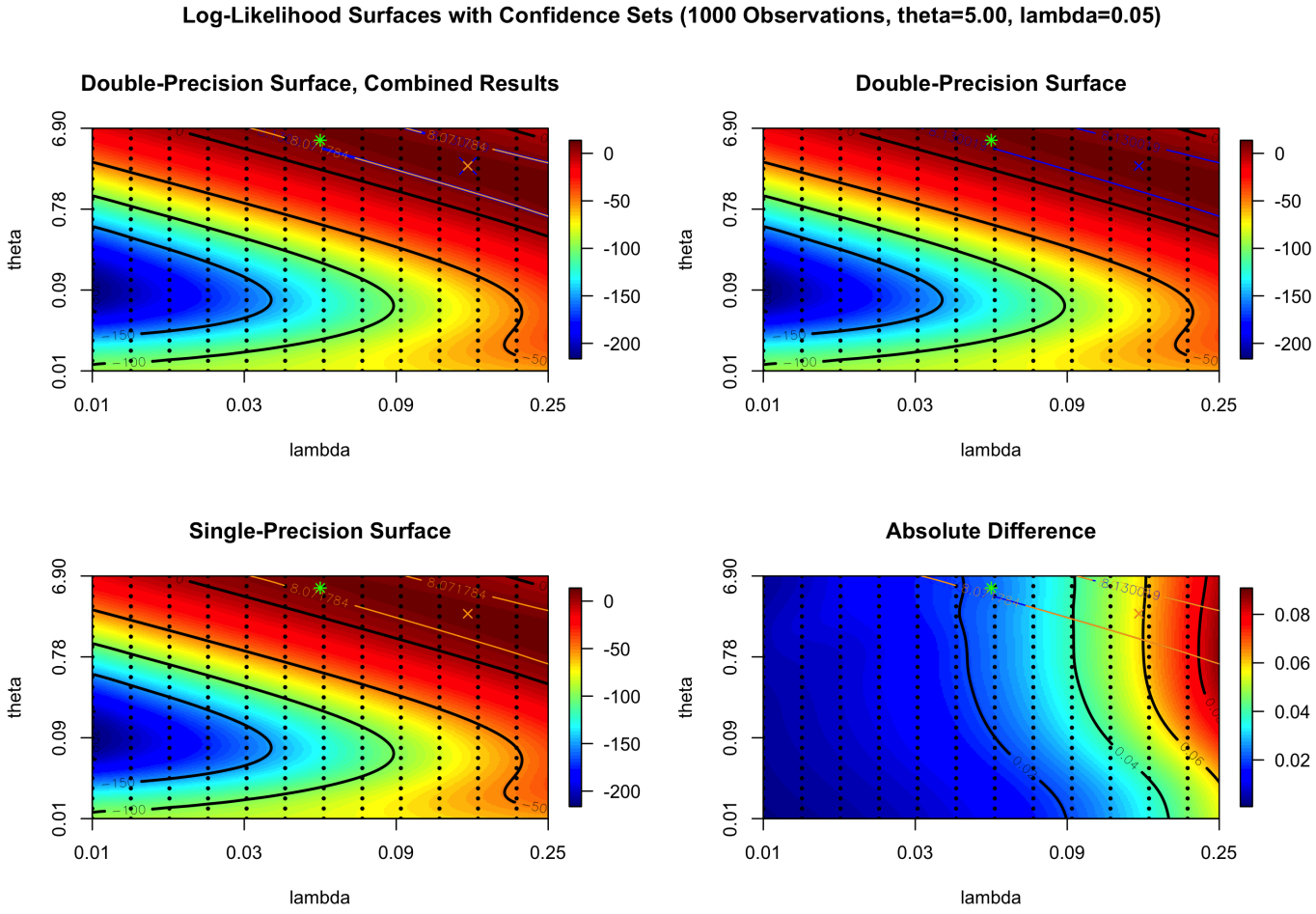


Figure 8: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.



(a)

Figure 9: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

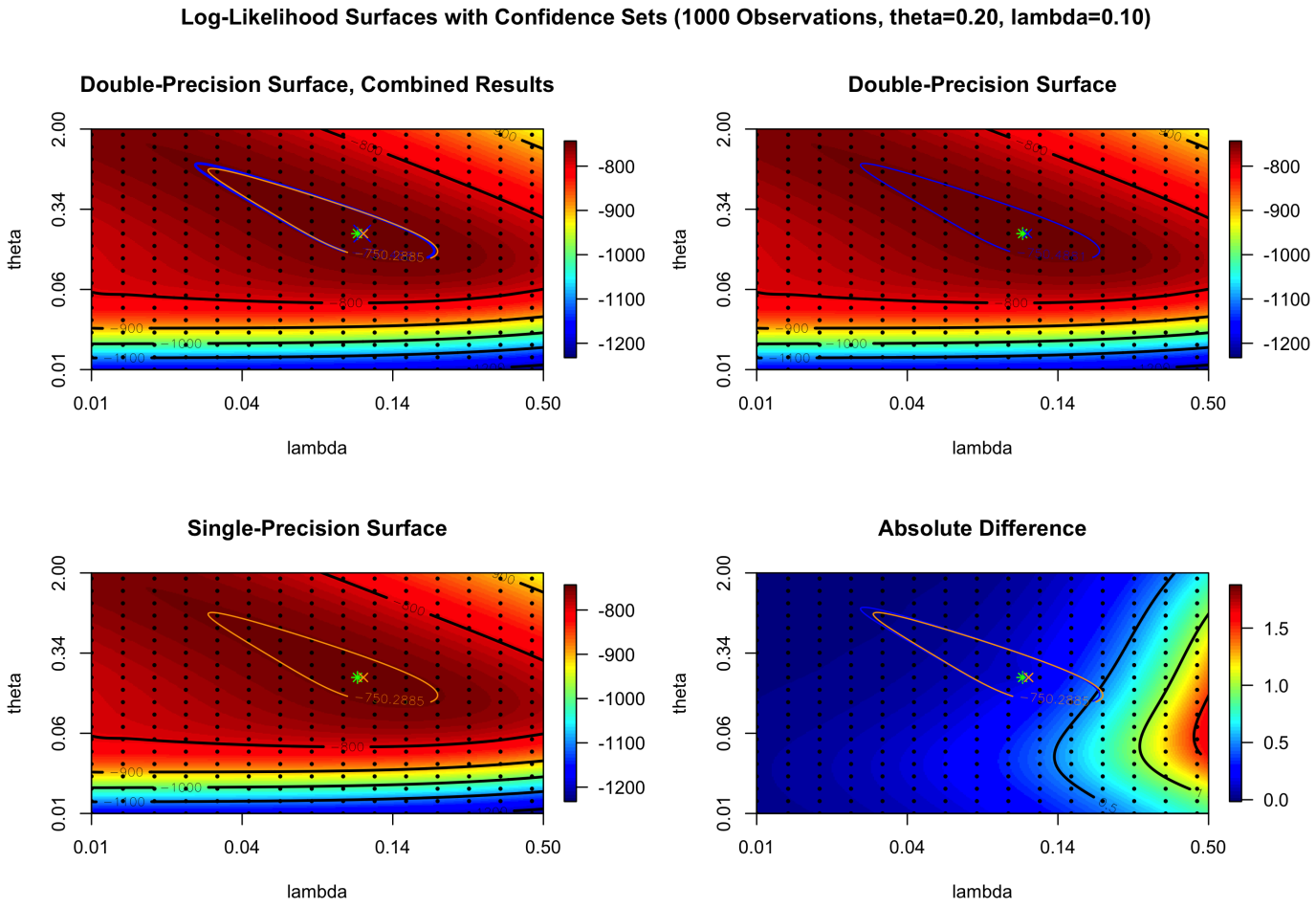


Figure 10: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

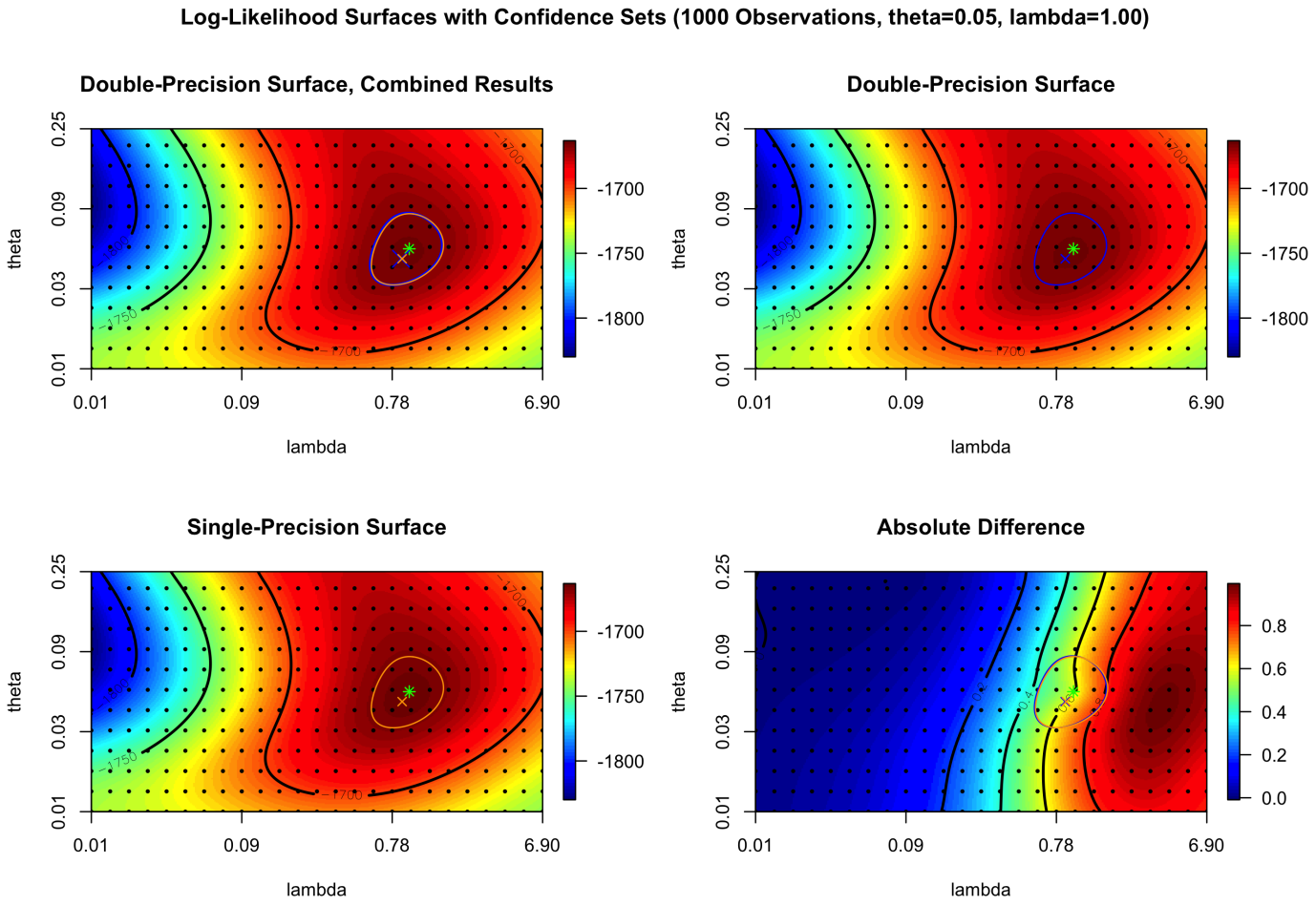


Figure 11: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

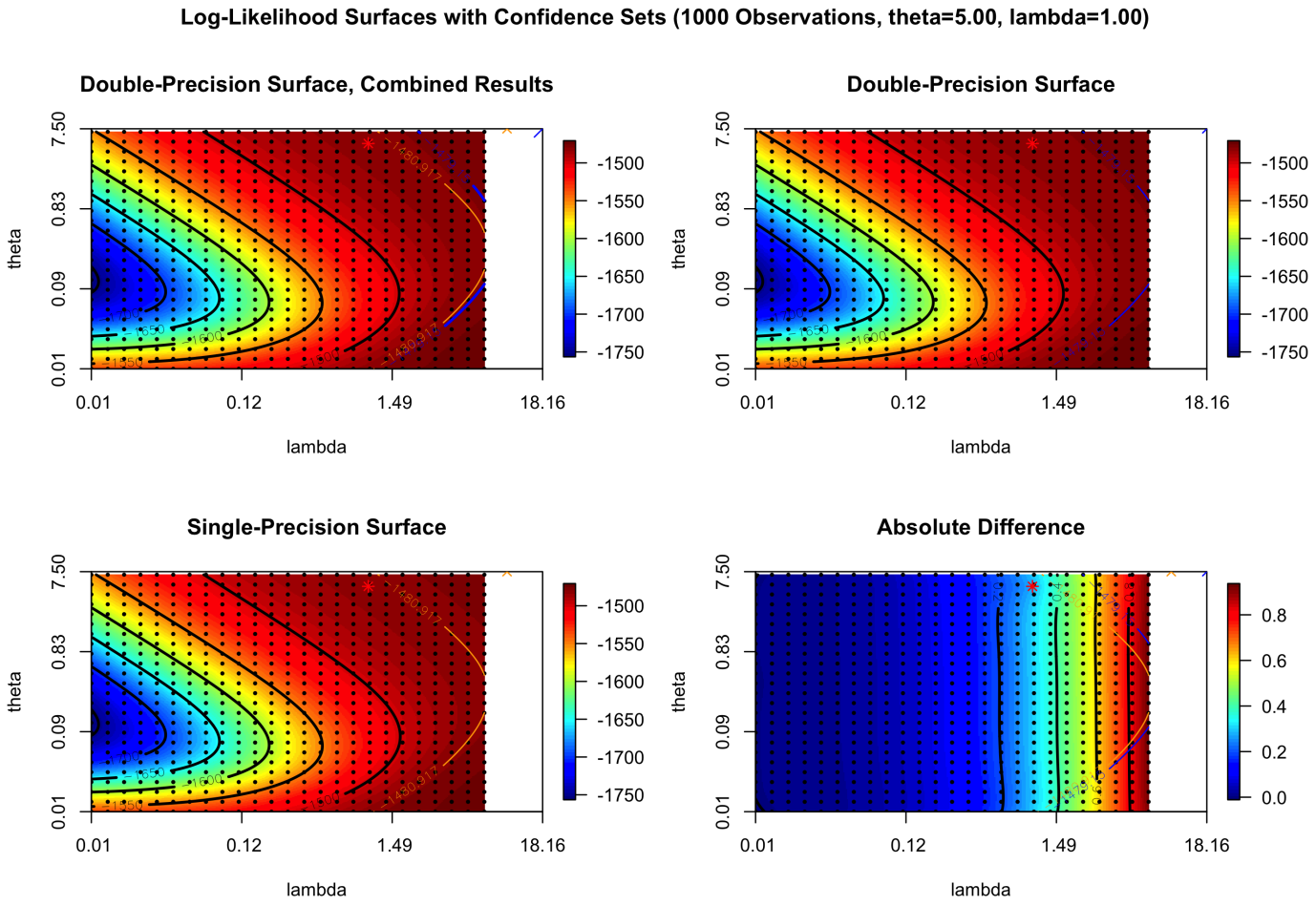


Figure 12: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

ever, it is clear from Figure 1 that at least the laptop GPU, the NVIDIA GeForce 750M with 2048 MB memory, single precision performance is much better than double precision performance, performing approximately 6 or 7 times faster for 10,000 observations respectively with calculations involving a copy of the factored matrix or that are in place. This improvement for single over double precision continues to improve as the number of observations increases.

The speedups of the accelerated Cholesky decomposition translate into smaller speedups for the accelerated version of `mKrig` and the speedups are further reduced for the accelerated spatial workflow. This is because the Cholesky decomposition is only a portion of the calculations in `mKrig` and the spatial workflow, although it takes the majority of the computation time. In spite of this, the speedup for the example spatial problem workflow is 7.2 and 16.4 times for double and single precision respectively, using approximately 10,000 observations. Although the speedups themselves are not as high as for the Cholesky decomposition itself, the time savings is much larger. Compared to the default workflow computation time of 26 minutes 55 seconds, the accelerated workflows respectively took 3 minutes 44 seconds and 1 minute 38 seconds for double and single precision. This corresponds to over 20 minutes of time saved. Of course, for a workflow with more than 1 likelihood evaluations, the speedup would be greater due to the larger portion of computations spent performing Cholesky decompositions to evaluate the likelihood. For instance, if θ, the exponential covariance scale parameter were not treated as fixed, more likelihood evaluations would probably be required.

It is important to note that the Caldera node GPUs (two NVIDIA Tesla K20Xs), used in Paige et al. (2015) had much better double precision performance compared to the 2014 MacBook's NVIDIA GeForce GT 750M. While the single precision speed of the Cholesky decomposition is only (roughly) twice that of the double precision on Caldera, it is approximately eight times faster on the MacBook Pro. This is because the GeForce GT 750M GPU has eight times as many single precision Arithmetic Logic Units (ALUs) as double precision ALUs, whereas the Tesla K20X uses twice as many (Brodtkorb et al. 2013, Du et al. 2012). Hence, although the newest GPUs still perform faster when operating in single precision, enabling single precision likelihood calculations would be of most benefit for the many `fields` users without access to the latest, state-of-the-art GPUs.

The faster computation time induced by using single precision Cholesky decompositions also seems to not significantly affect the parameter optimization process. In all of our tests the true parameters were in the double precision 95 percent confidence set if and only if

they were also in the equivalent confidence set calculated using single precision. We believe the parameters used when generating the data in the likelihood calculations reflect fairly extreme and relatively common circumstances alike when accounting for the fact that the data's spatial domain is the unit square.

Although the single precision Cholesky decomposition resulted in a numerically singular matrix in cases where the double precision decomposition did not, the parameter values and distances where this occurred are not likely to occur in many real-world applications. The spatial domain of the simulated datasets has diameter at most $\sqrt{2}$, which is approximately 7,000 times smaller than the problematic spatial scale parameter singular threshold, at least 140 million times larger than the distances resulting in numerical singularity of the covariance matrix, and at least 1,400 times larger than problematic values of the Kriging spatial smoothing parameter, λ. Even in the case where points are too close together, causing the covariance matrix to be numerically singular, this can in many cases be easily prevented by removing any conflicting observations.

While exploiting heterogeneous architectures (those with CPUs and coprocessors such as GPUs) is important, there is an inherent limit to what the Kriging algorithm can accomplish if it must use an $\mathcal{O}(n^3)$ operation for n spatial observations. `R` packages such as `latticeKrig` (Nychka et al. 2014b) induce sparsity in the matrix to perform the Cholesky decomposition by using the precision matrix in combination with compactly supported covariance functions (Nychka et al. 2014a). MAGMA is not best suited for cases like these where the matrix operated on is sparse, as it is designed primarily to accelerate dense matrix operations (Agullo et al. 2009). It is therefore important to incorporate accelerated sparse Cholesky decomposition algorithms for heterogeneous systems into spatial statistics packages that use sparse matrices in the future.

References

Agullo, Emmanuel, Jim Demmel, Jack Dongarra, Bilel Hadri, Jakub Kurzak, Julien Langou, Hatem Ltaief, Piotr Luszczek, and Stanimire Tomov (2009), "Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects." In *Journal of Physics: Conference Series*, volume 180, 012037, IOP Publishing.

Brodtkorb, André R, Trond R Hagen, and Martin L Sætra (2013), "Graphics processing unit

(GPU) programming strategies and trends in gpu computing." *Journal of Parallel and Distributed Computing*, 73, 4–13.

Cressie, Noel and Gardar Johannesson (2008), "Fixed rank kriging for very large spatial data sets." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70, 209–226.

Du, Peng, Rick Weber, Piotr Luszczek, Stanimire Tomov, Gregory Peterson, and Jack Dongarra (2012), "From cuda to opencl: Towards a performance-portable solution for multi-platform gpu programming." *Parallel Computing*, 38, 391–407.

Katzfuss, Matthias and Noel Cressie (2012), "Bayesian hierarchical spatio-temporal smoothing for very large datasets." *Environmetrics*, 23, 94–107.

Nickolls, John and William J Dally (2010), "The GPU computing era." *IEEE micro*, 30, 56–69.

Nychka, Douglas, Soutir Bandyopadhyay, Dorit Hammerling, Finn Lindgren, and Stephan Sain (2014a), "A multi-resolution gaussian process model for the analysis of large spatial data sets." *Journal of Computational and Graphical Statistics*, 00–00.

Nychka, Douglas, Dorit Hammerling, Stephan Sain, and Nathan Lenssen (2014b), *LatticeKrig: Multiresolution Kriging based on Markov random fields*. URL `http://CRAN.R-project.org/package=LatticeKrig`. R package version 3.4.

Paige, John, Douglas Nycka, Isaac Lyngaas, Srinath Vadlamani, and Dorit Hammerling (2015), "Incorporating MAGMA into the 'fields' spatial statistics package." Technical report, The National Center for Atmospheric Research.

Stein, Michael L (2014), "Limitations on low rank approximations for covariance matrices of spatial data." *Spatial Statistics*, 8, 1–19.

Tomov, S, J Dongarra, V Volkov, and J Demmel (2009), "MAGMA library." *Univ. of Tennessee and Univ. of California, Knoxville, TN, and Berkeley, CA*.

Acknowledgements

This work would not have been possible without the help of Isaac Lyngaas, a PhD student at Florida State University, and Dr. Srinath Vadlamani, a Computational Scientist at ParaTools, Inc., during its initial stages. This work was partially completed while Paige and Lyngaas were interns in Summer Internships in Parallel Computational Sciences (SIParCS). SIParCS is a program at the National Center for Atmospheric Research (NCAR), which is sponsored by the National Science Foundation. We would like to thank the Consulting Services Group at NCAR. In particular, Siddhartha Ghosh answered several of our technical questions on using `R` in the Yellowstone supercomputing environment. In addition, Tim Hoar, an Associate Scientist at NCAR, helped install and link MAGMA with `R`.

5 Appendix

5.1 Installation

5.1.1 Installing MAGMA and Accelerated `fields` on a Mac

The following instructions are for installing MAGMA and our accelerated `fields` methods on a mid-2014 MacBook Pro laptop with a NVIDIA DeForce GT 750M GPU with 3 GB VRAM and CUDA compute capability 3.0. The computer was also fitted with a 2.8 GHz Intel Core i7 quad=core CPU with 16 GB, 1.6 GHz DDR3 RAM.

1. Install Macports, if necessary (`https://www.macports.org/install.php`)
2. Install the gcc 4.8 compiler suite using steps 3-5, if necessary. Otherwise, skip to step 6.
3. Update Macports using the command: "`sudo port selfupdate`"
4. Run command: "`sudo port install gcc48 +gfortran`"
5. After the above command is complete, `gcc`, `g++`, and `gfortran` compilers should be installed to "`/opt/local/bin/`". Run:

```
port select --list gcc
cd /opt/local/bin/
sudo port select --set gcc mp-gcc48
gcc -v
```

substituting `/opt/local/bin/` as appropriate to wherever Macports installed mp-gcc48 on your computer. Your output for the first line should include "`mp-gcc48`", and the output for the last line should include "`gcc version 4.8.x`".

6. Install OpenBLAS on your Mac using Macports with the command "`port install openblas`". OpenBLAS should now be installed in `/opt/local` under `lib` and `include` directories.

7. Check your GPU CUDA compute capability from `https://developer.nvidia.com/cuda-gpus`. If your GPU's compute capability is below 2.0, make sure to get a version of CUDA before 6.5. Download CUDA if necessary from `https://developer.nvidia.com/cuda-downloads`. This installation guide uses CUDA 7.0.36, but it might be possible to use another version of CUDA with similar instructions. If you already have CUDA, skip to step 9.

8. Follow NCIDIA's CUDA Mac OS X installation instructions ("Prerequisites," "Installation," and "Verification" sections) available at `http://doc.nvidia.com/cuda/cuda-getting-started-guide-for-mac-os-x/#axzz3XJqJPRHg`. Make sure to install the CUDA Driver as well as the CUDA Toolkit, and to setup traditional cuda-gdb permissions at the end of the installation. It is also strongly recommended to install the local samples to run tests to make sure CUDA was installed correctly.

9. Create a `.bash_profile` document in your home directory, if necessary, and include in it the line "`source ~/.bashrc`" (run "`ls -a`" in Terminal to show hidden files starting with a ".")

10. Now create a "`.bashrc`" file in your home directory to personalize your terminal settings every time you open a terminal window. As suggested in CUDA's installation instructions, make sure the following lines are in `.bashrc`:

```
export PATH=/Developer/NVIDIA/CUDA-7.0/bin:$PATH
export DYLD_LIBRARY_PATH=/usr/local/cuda/lib/:\
/Developer/NVIDIA/CUDA-7.0/lib:$DYLD_LIBRARY_PATH
export CUDADIR=/Developer/NVIDIA/CUDA-7.0
export OPENBLASDIR=/opt/local
```

    Note: make sure to adjust the paths to point to your specific version of CUDA and OpenBLAS.

11. If you have a `~/.profile` file (possibly created by installing Macports) make sure to include the line:

`source ~/.profile`

in `.bashrc`.

12. Open a new terminal window for the changes to come into effect

13. Download MAGMA version 1.6.1 from `http://icl.cs.utk.edu/magma/software/index.html`. These instructions may also work for other versions of MAGMA, but they assume MAGMA version 1.6.1.

14. Untar the .tar file and move it to your home directory. Modify "`make.inc.macos`" in your MAGMA directory to include the following lines:

```
GPU_TARGET = Kepler
CFLAGS    = -m64 -O3 $(FPIC) -DADD_ -Wall -fopenmp
LDFLAGS   = -m64     $(FPIC) -fopenmp
LIB       = -framework Accelerate -lopenblas -lcublas -lcudart -lstdc++ -lm

-include make.check-openblas

LIBDIR    = -L$(CUDADIR)/lib \
            -L$(OPENBLASDIR)/lib

INC       = -I$(CUDADIR)/include \
            -I$(OPENBLASDIR)/include
```

    In other words, set the "`GPU_TARGET`" to "`Kepler`" (although this will depend on your GPUs CUDA compute capability), add the "`-lopenblas`" tag to `LIB`, add "`-include make.check-openblas`", and add "`-L$(OPENBLASDIR)/lib`" to `LIB_DIR`. Note that if your GPU has CUDA compute capability 2.x substitute `Fermi` for `Kepler` and if it is 1.x substitude `Tesla` for `Kepler` (although that will only work for CUDA version < 6.5.

15. Copy `make.macos` and rename the resulting file "`make.inc`" in the MAGMA directory

16. Still in the MAGMA directory, run "`make`" to build MAGMA (or, if this is not your first time running the command, run "`make clean`" first).

17. After MAGMA is installed, run:

```
./testing_dpotrf -c --ngpu 1
./testing_dpotrf_m -c --ngpu 1
./testing_spotrf -c --ngpu 1
./testing_spotrf_m -c --ngpu 1
```

in MAGMA's testing directory. Note that the `-h` option gives help for the different options you can pass to the testing functions. If the commands are both successful, MAGMA should be successfully installed on your Mac.

5.1.2 Installing `fieldsMAGMA` on a Mac

MAGMA is required for these instructions, which show how to use MAGMA's Cholesky decomposition from `R` and how to install `fieldsMAGMA`.

1. If either ∼`/.R` or ∼`/.R/Makevars` does not exist (run "`ls -a`" in Terminal to show hidden files starting with a "."), create them and edit `Makevars`. Add the following lines to `Makevars`:

   ```
   CC=/opt/local/bin/gcc
   CXX=/opt/local/bin/gcc
   F77=/opt/local/bin/gfortran
   FC=/opt/local/bin/gfortran
   PKG_CFLAGS= -I/Users/USERNAME/magma-1.6.1/include -I/opt/local/include \
   -I/usr/local/cuda/include -I/Developer/NVIDIA/CUDA-7.0/include -DHAVE_CUBLAS \
   -DADD_ -fopenmp -std=c99
   PKG_LIBS=/Users/USERNAME/magma-1.6.1/lib/libmagma.a -L/opt/local/lib \
   -lopenblas -L/usr/local/cuda/lib/ -L/Developer/NVIDIA/CUDA-7.0/lib -lcudart \
   -lcublas -fopenmp
   ```

   substituting file paths for those on your specific system as necessary and `USERNAME` for your user name.

2. Download the `fieldsMAGMA` package from `https://bitbucket.org/jpaige/rpackages` by pressing the "Downloads" button on the left side of the window.

3. `cd` into the downloaded repository on the command line and run:

   ```
   R CMD BUILD fieldsMAGMA
   R CMD INSTALL fieldsMAGMA_1.0.tar.gz
   ```

   making sure to substitute "1.0" for the appropriate version of fieldsMAGMA.

4. If you are using RStudio or Rconsole, you may find an error when loading "`fieldsMAGMA`" similar to:

   ```
   Error in dyn.load(file, DLLpath = DLLpath, ...) :
     unable to load shared object '/Users/jpaige/Library/R/3.1/library/
     fieldsMAGMA/libs/fieldsMAGMA.so':
     dlopen(/Users/jpaige/Library/R/3.1/library/fieldsMAGMA/libs/
     fieldsMAGMA.so, 6): Library not loaded: @rpath/libcudart.7.0.dylib
   ```

```
  Referenced from: /Users/jpaige/Library/R/3.1/library/fieldsMAGMA/
  libs/fieldsMAGMA.so
  Reason: image not found
Error: package or namespace load failed for fieldsMAGMA
```

In this case, the "`@rpath`" loading path variable is incorrect, and it is likely due to a problem with RStudio or Rconsole since it loads correctly when running from Terminal. One way to fix this error is to load `fieldsMAGMA` when running `R` from Terminal instead. If you do not want to have to use `fieldsMAGMA` from the command line each time, an alternative is to `cd` to the directory enclosing "`fieldsMAGMA.so`" and run:

```
install_name_tool -change "@rpath/libcudart.7.0.dylib" \
"/Developer/NVIDIA/CUDA-7.0/lib/libcudart.7.0.dylib" fieldsMAGMA.so
```

where the path names should represent the specific path names for you system and installation. You may need to run the equivalent command for `libcublas` as well. The `otool` command with the `-L` option shows the paths of the libraries linked to the file taken as its argument. You should now be able to load the `fieldsMAGMA` package in `R`.

5. Test to make sure everything runs correctly using the steps from Paige et al. (2015). Begin an `R` session and run the following lines of code to check you have installed the accelerated code correctly:

```
#get magmaChol MAGMA-accelerated Cholesky decomposition
library(fieldsMAGMA)

#This function generates an n x n positive definite matrix
getMatForChol = function(n) {
  #get a random matrix
  A = matrix(rnorm(n^2), n, n)

  # A*A^T is positive definite
  return(A %*% t(A))
}

Sigma = getMatForChol(5)
U = magmaChol(Sigma)
SigmaEst = t(U)%*%U

# The following printed matrices should be the same:
print(Sigma)
print(SigmaEst)

# See how much faster MAGMA is:
```

```
Sigma = getMatForChol(2000)
system.time(U1 <- magmaChol(Sigma))[3]
system.time(U2 <- chol(Sigma))[3]

#check to make sure MAGMA Cholesky agrees with default
max(abs(U1 - U2))
```

In order to test using MAGMA to perform the Cholesky decomposition with n GPUs at once, substitute "`U = magmaChol(Sigma, nGPUs=n)`" for "`U = magmaChol(Sigma)`". Run `?magmaChol` in `R` to see its various arguments that can be set to improve or customize its performance.

6. If both matrices (`Sigma` and `SigmaEst`) printed in the above code are the same, then the code should be working correctly, and you can use `fieldsMAGMA` in `R`.

5.2 Supplemental Single Precision Likelihood Results

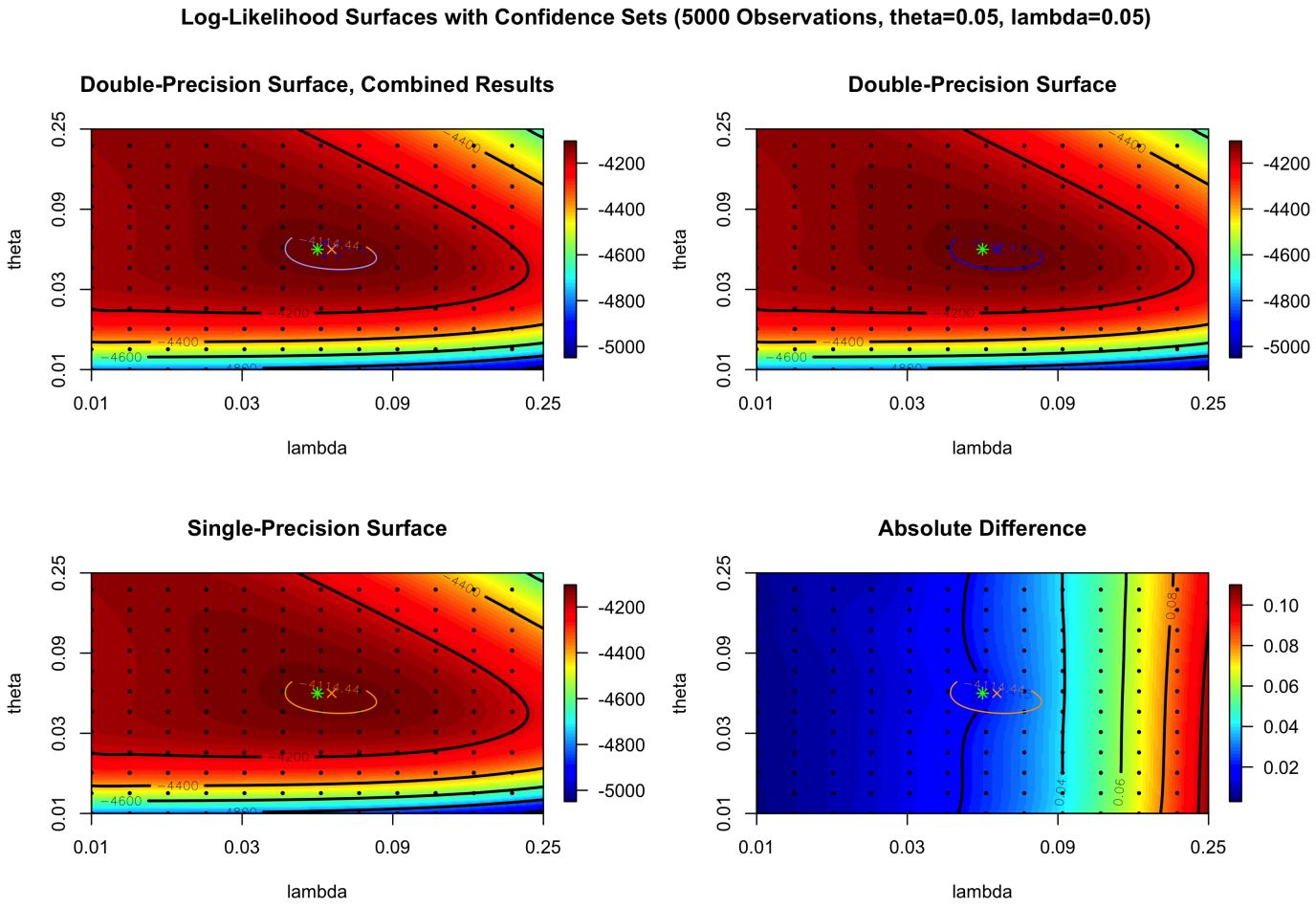
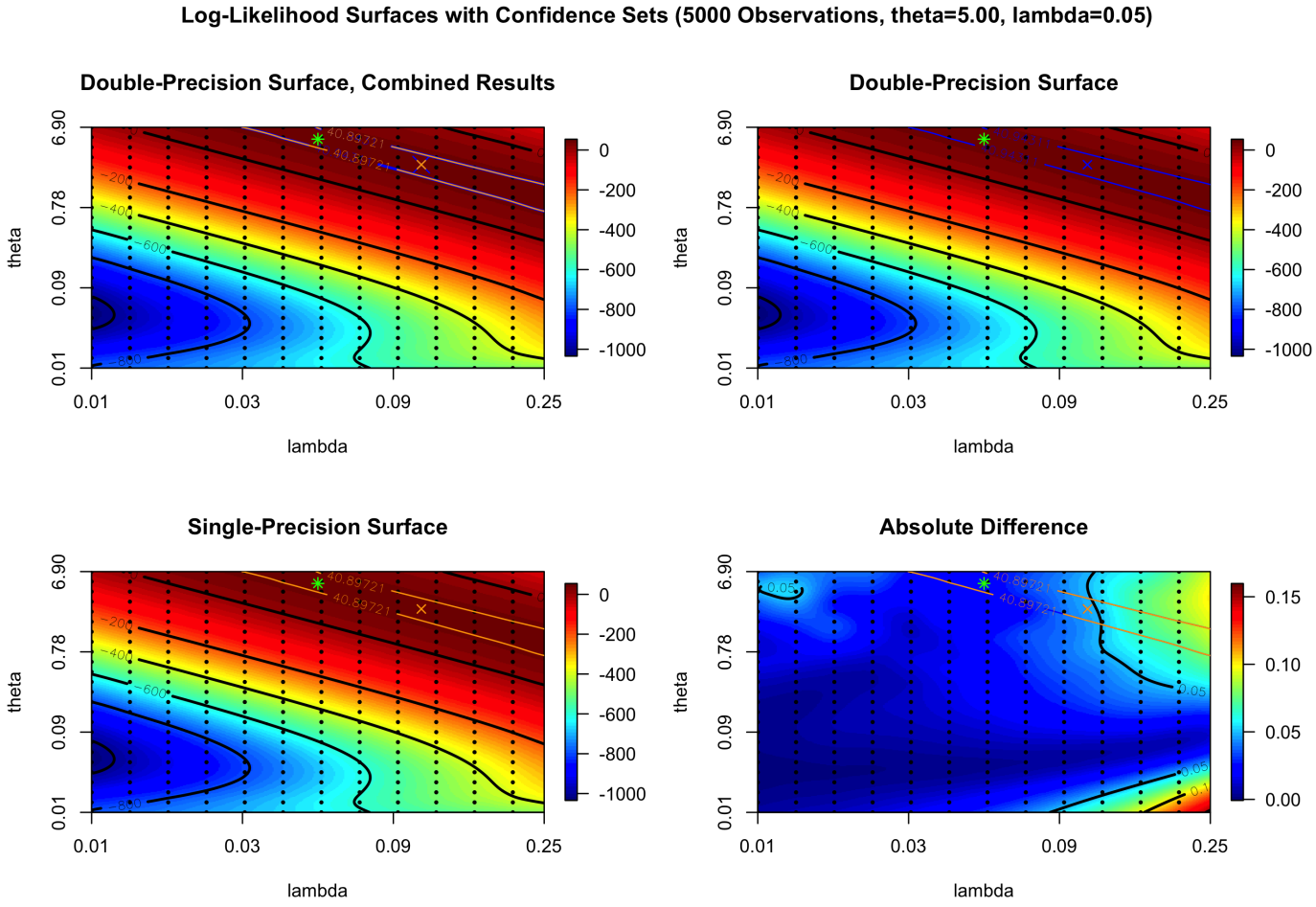


Figure 13: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.



(a)

Figure 14: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

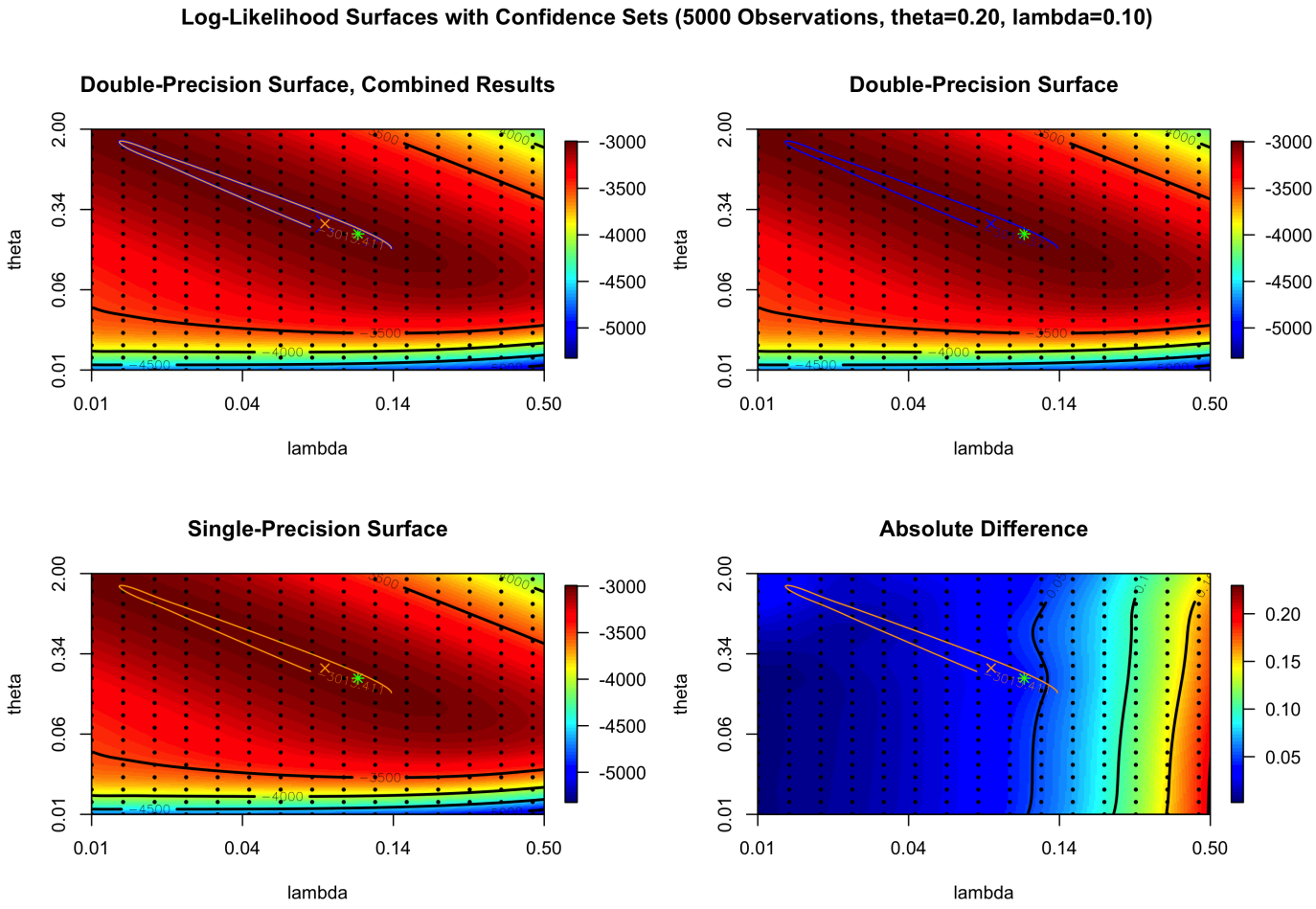


Figure 15: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

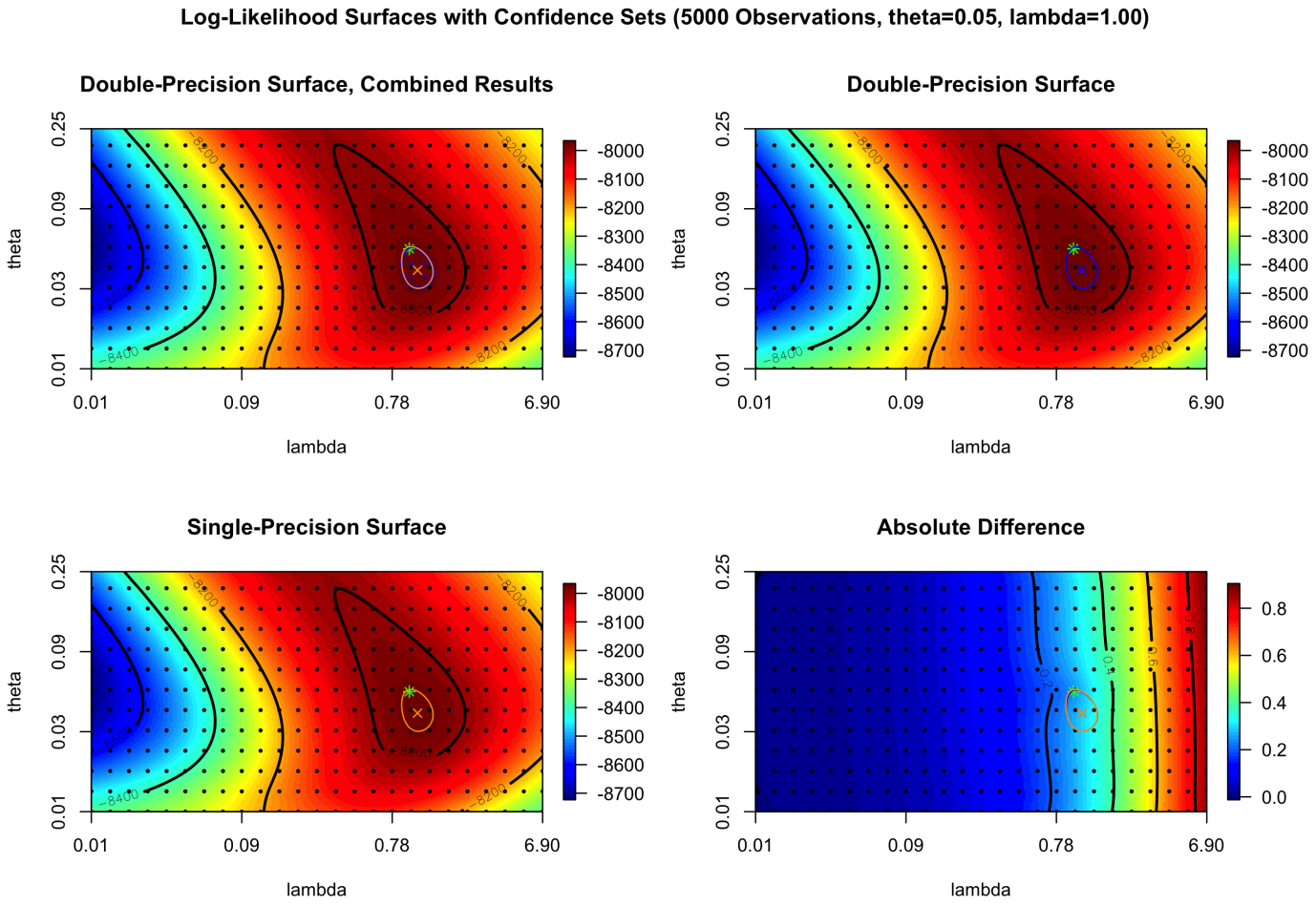


Figure 16: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

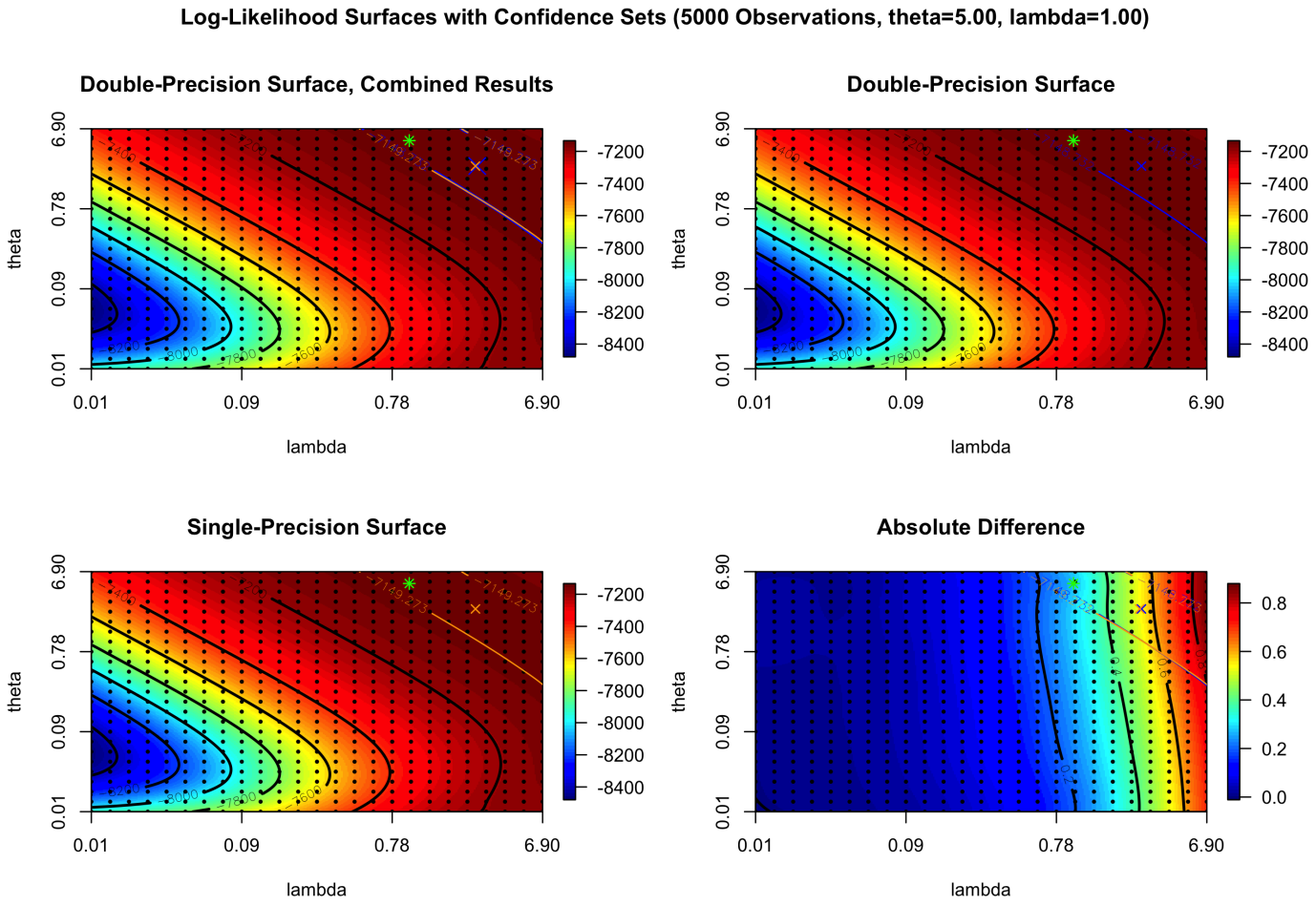


Figure 17: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

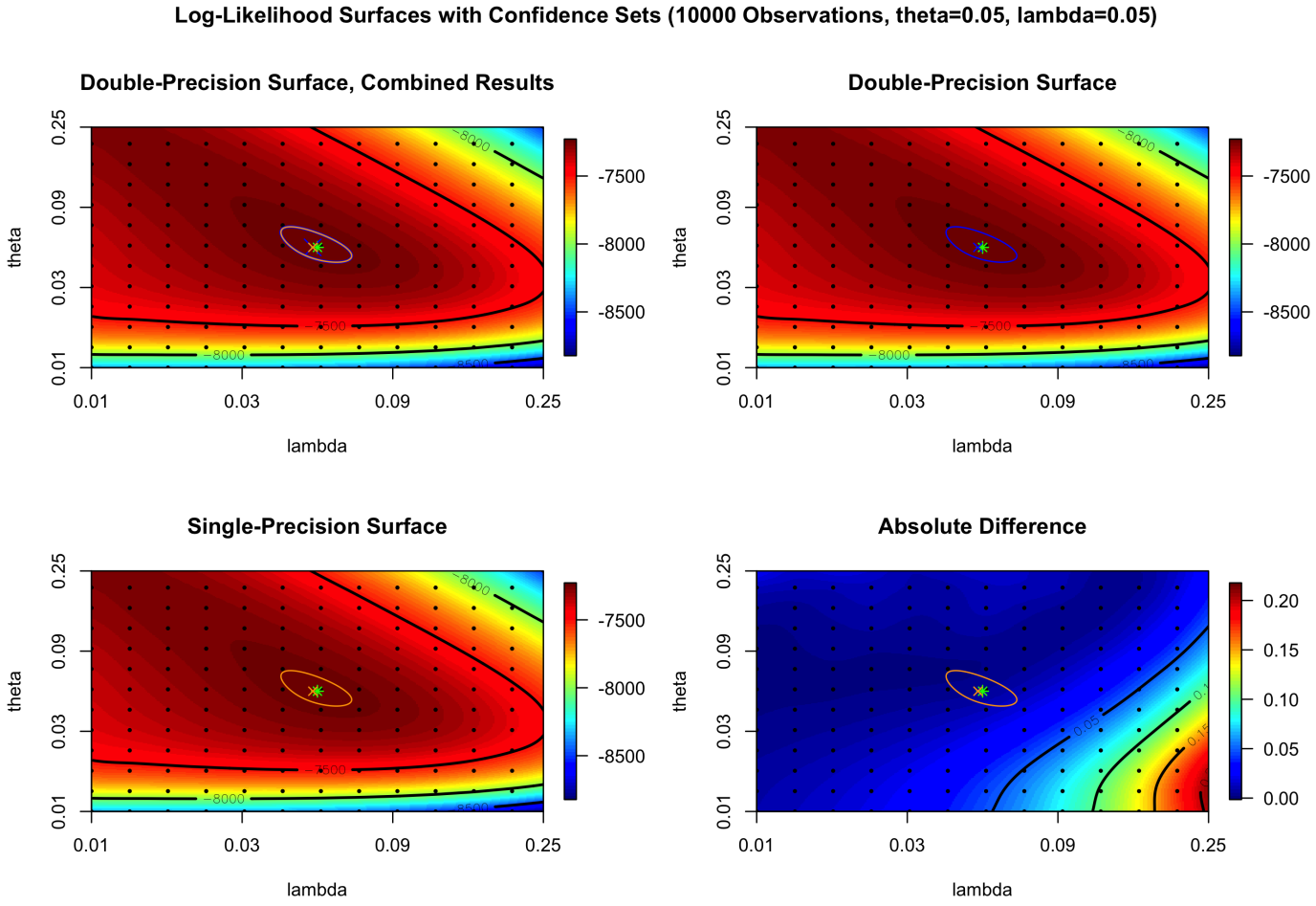
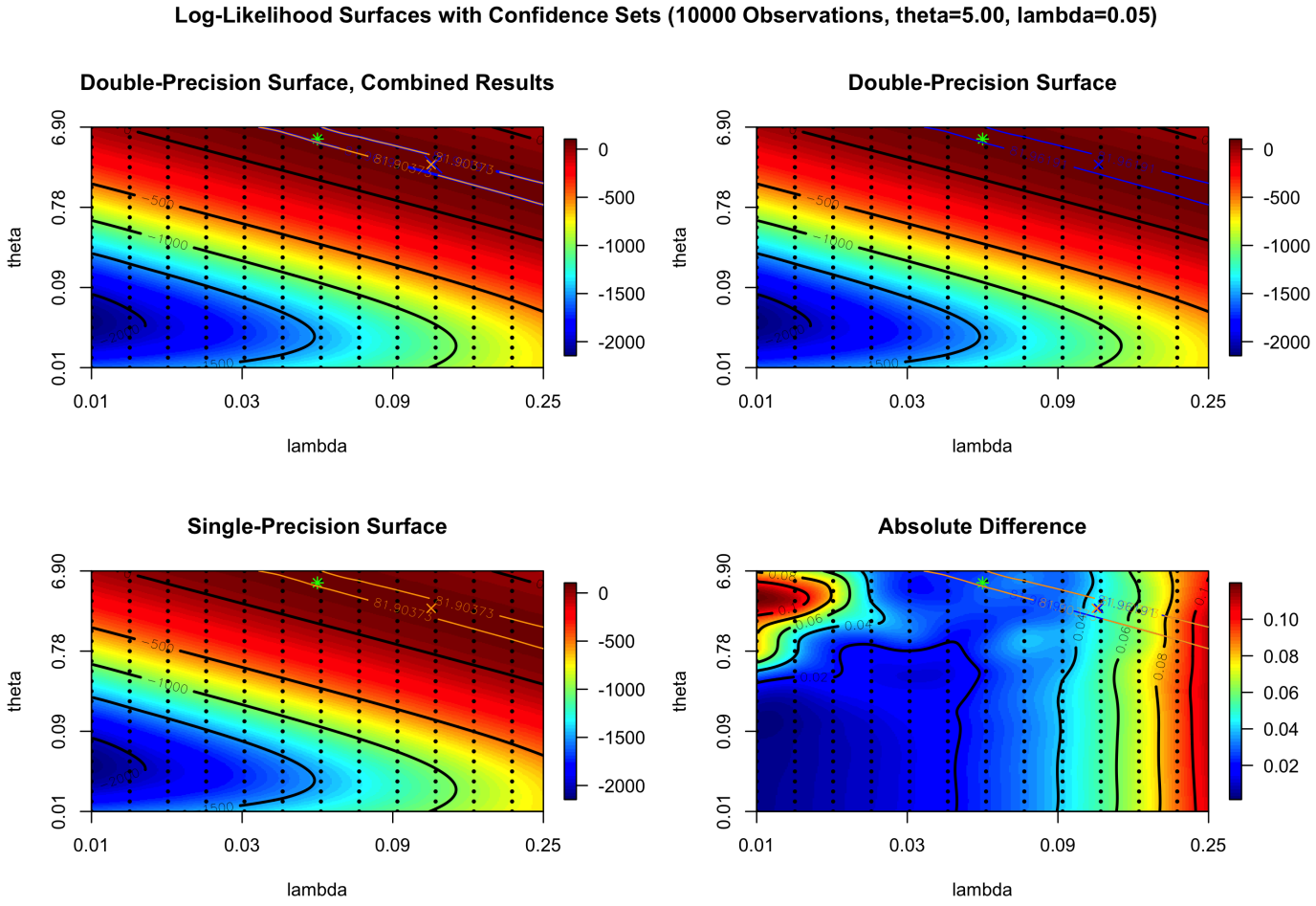


Figure 18: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.



(a)

Figure 19: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

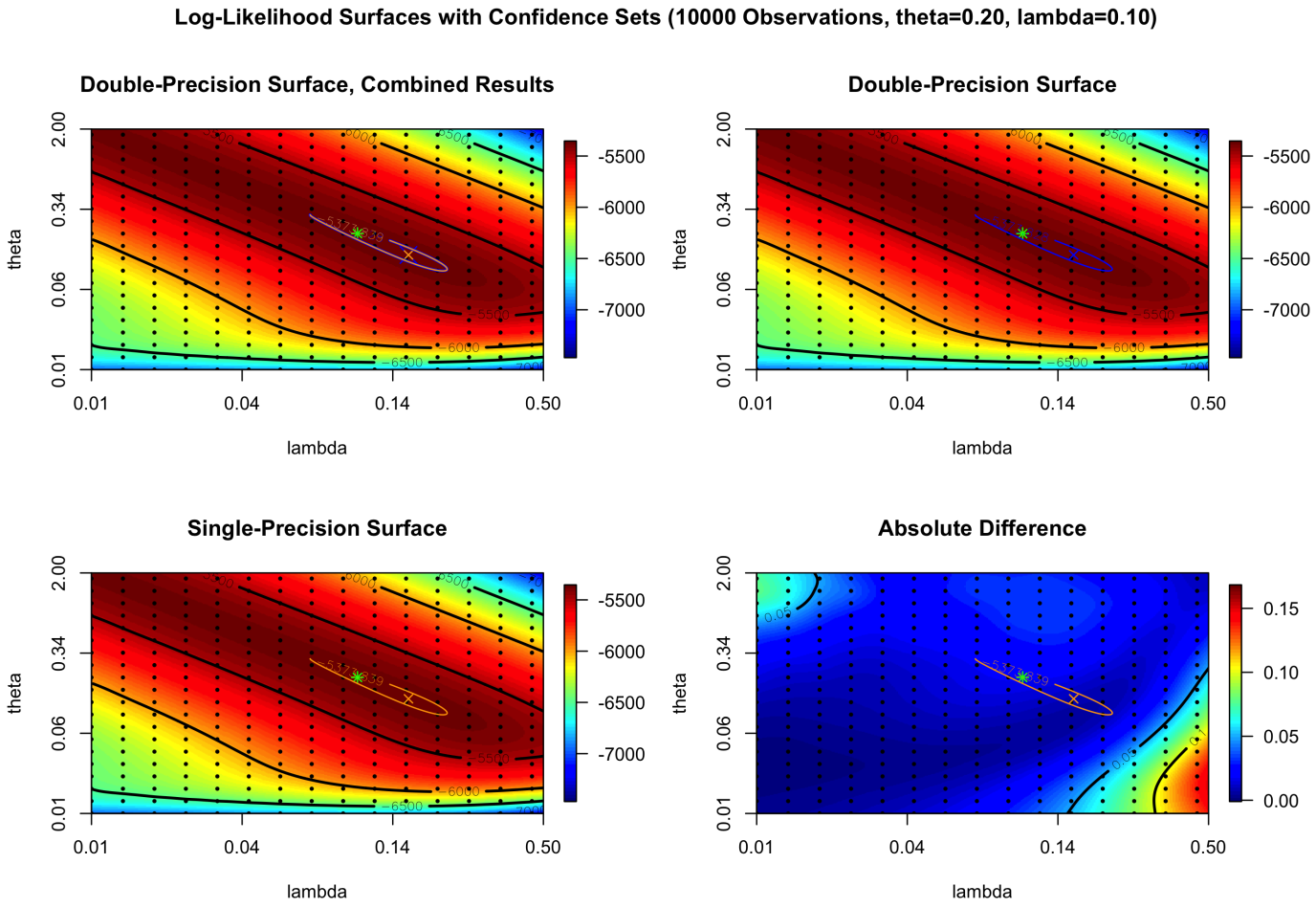


Figure 20: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

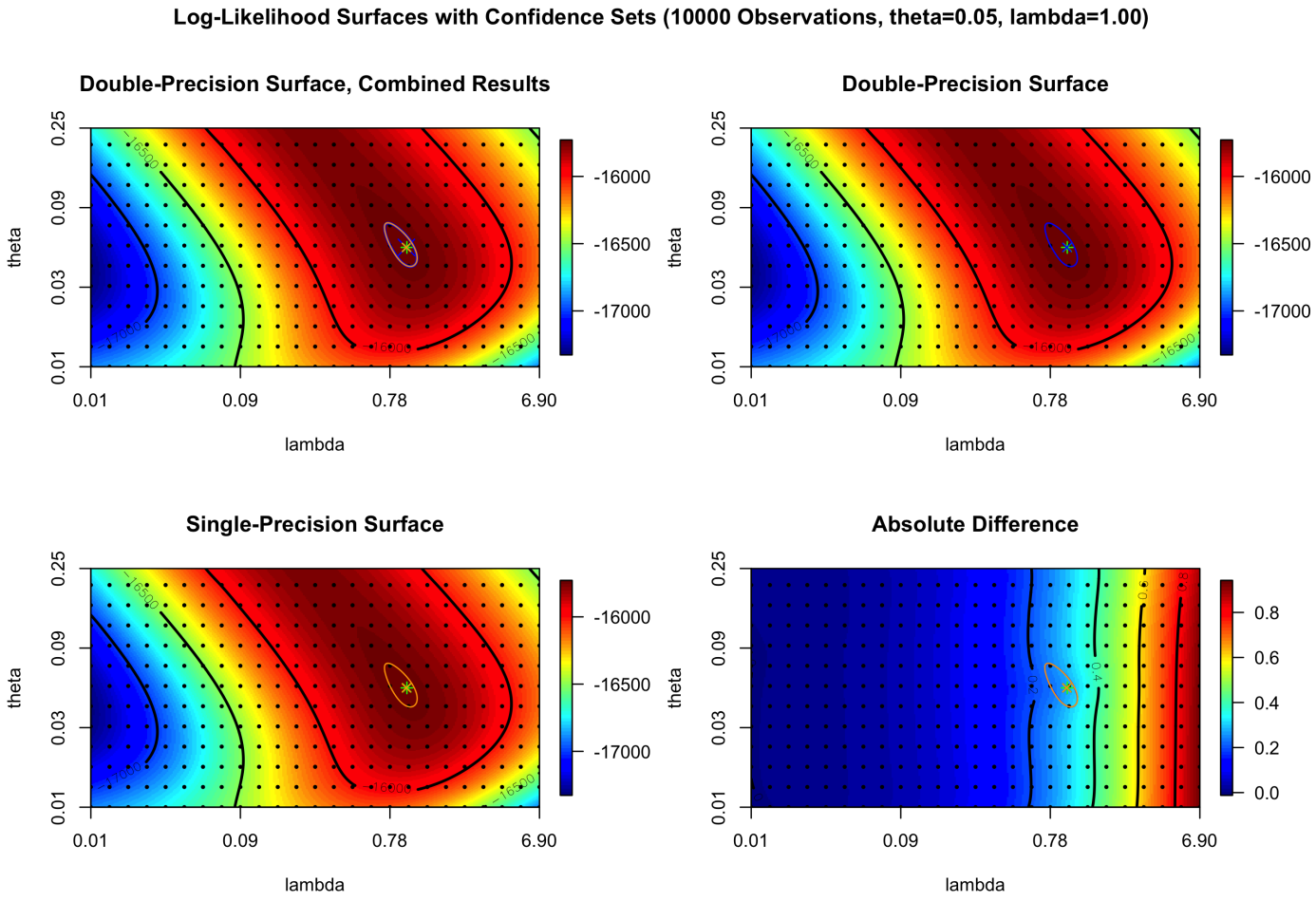


Figure 21: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.

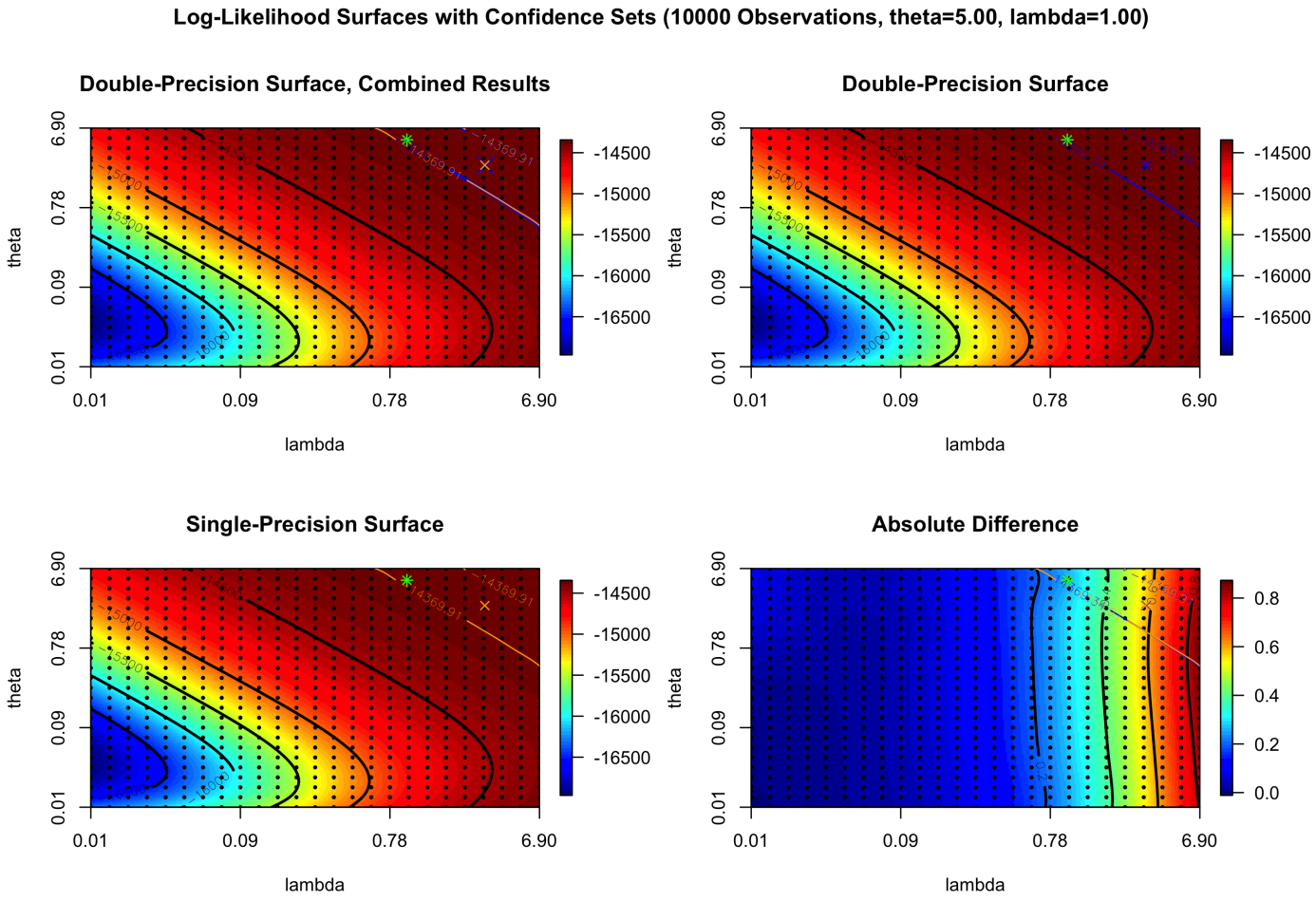


Figure 22: The above plots show thin plate spline approximations to the log-likelihood surface for simulated observations as calculated in single and double precision as well as a thin plate spline approximation to the absolute difference between the surfaces. The log-likelihood surfaces are sampled for parameter values plotted as black dots, where θ is the exponential covariance scale parameter and λ is the Kriging smoothing parameter. 95 percent confidence sets (contours) for the true parameters used to generate the data are shown along with MLE parameter estimates (crosses) in orange (single precision) and blue (double precision). The true parameters are shown as a green star. For 1,000, 5,000, and 10,000 observations, observations are generated 10, 2, and 1 times independently, and the depicted surfaces correspond to the replication where the absolute difference in single and double precision MLE log-likelihoods is largest.