Classical and Quasi-Newton Methods for a Meteorological Parameters Prediction Boundary Value Problem

Ioannis Famelis, Georgios Galanis, Matthias Ehrhardt, and Dimitrios Triantafyllou

1 Department of Electronics Engineering, TEI of Athens, Greece
2 Section of Mathematics, Hellenic Naval Academy, Greece
3 Fachbereich C - Mathematik und Naturwissenschaften, Angewandte Mathematik und Numerische Analysis, Bergische Universität Wuppertal, Germany

Published online: 1 Nov. 2014

Abstract: We study the numerical solution of a Boundary Value problem for second order quadratic differential equations which arises in the numerical prediction of meteorological parameters. In the present work, we use finite differences and focus on the numerical solution of the resulting nonlinear system. More precisely, we apply classical Newton’s and Quasi-Newton methods paying attention to the special sparse form of the Jacobian matrix and modify appropriately the LU factorization in order to reduce significantly the required floating point operations. Furthermore, we implement and study in depth the behavior of all the proposed procedures in respect of their accuracy, stability and complexity, using data from South East Mediterranean Sea. All the methods are tested with a variety of initial values and their performance is presented and discussed leading to interesting results on the sensitivity of the selected starting point.

Keywords: Information Geometry, Boundary Value Problem, Finite differences, Newton method, Quasi Newton methods, Numerical solution of nonlinear systems, sparse matrix

1 The Physical Problem and Information Geometry

In the recent years, the need of accurate local predictions of environmental parameters has increased significantly as a result of the large number of related social and commercial activities e.g. climate change, renewable energy production, transportation, marine pollution, ship safety. This has led to the activation of numerous research and operational centers that develop high quality scientific tools able to provide reliable environmental predictions.

In the study of such problems one may use in site or remote sensing (e.g. satellite) observations or numerical prediction models that solve the basic equations governing the atmospheric and wave evolution based on a numerical analysis approach. Numerical weather and wave forecasting models have been proved successful for the simulation of the general environmental conditions on global or intermediate scale. However, when focusing on local conditions usually systematic errors appear [1, 2, 3, 4, 5]. Such problems are multi-parametric and several different issues are involved like the strong dependence on the initial and lateral conditions, the inability to capture sub-scale phenomena, the parametrization of certain atmospheric or wave procedures.

To deal with these difficulties one may increase the model resolution, but it is still an open question if this leads to a considerable improvement of the forecast quality. Even if this is true, it also results to an considerable increase of the computational costs.

Alternatively, we can improve the initial conditions based on assimilation systems. Corrected analysis fields are provided to the numerical models based on available observations from meteorological stations, satellites or other sources. Post process algorithms are used for the improvement of the final results based on statistical models (MOS methods, Neural networks, Kalman filters). One of the key issues is the appropriate estimation of the "distance" between two distributions or data sets. The classical treatment of such problems is usually based on

* Corresponding author e-mail: ifamelis@teiath.gr
regression techniques - least squares methods. However, such an approach carries the assumption that the data processed belongs to an Euclidian - finite dimensional space.

For example, in the case of Kalman filters, the evolution in time of an unknown process \( x_t \) is described by the system equation:

\[
x_t = F_t \cdot x_{t-1} + w_t.
\]

Here a known process \( y_t \) is coupled to \( x_t \) by the observation equation

\[
y_t = H_t \cdot x_t + v_t.
\]

The filter is based on the minimization of the covariance matrix \( E(x_t x'_t) \) of \( x_t \).

However, the following question need to be addressed: Is the distance/cost-function correctly estimated by means of classical Euclidean Geometry tools?

Recently, optimization techniques in the framework of a relatively new branch of mathematics, the Information Geometry (IG) \([6,7,8]\), are employed. Methods and techniques of non-Euclidean geometry to stochastic processes provide the tools to define a notion of distance between two probability distributions or two data sets. This affects crucially the cost function estimation. IG shows that the use of Euclidean/flat geometry techniques is false in general, and provides a theoretical recipe to avoid such simplifications. Families of probability distributions are recognized as manifolds on which geometrical entities such as Riemannian metrics, distances, curvature and affine connections can be naturally introduced.

More precisely, a \( n \)-dimensional statistical manifold is a family of probability distributions

\[
S = \{ p = p(x; \xi) \mid \xi = [\xi_1, \xi_2, \ldots, \xi_n] \in \Xi \},
\]

where each element may be parameterized using \( n \) real valued variables in an open subset \( \Xi \) of \( \mathbb{R}^n \) while the mapping \( \xi \rightarrow p_\xi \) is injective and smooth. The geometrical framework in a statistical manifold is characterized by the information matrix which at a point \( \xi \) is a \( n \times n \) matrix \( G(\xi) = [g_{ij}(\xi)] \), with elements

\[
g_{ij}(\xi) = \mathbb{E}_{\xi}[\partial_i \ell(x; \xi) \partial_j \ell(x; \xi)] = \int \partial_i \ell(x; \xi) \partial_j \ell(x; \xi) p(x; \xi) dx,
\]

where \( \ell(x; \xi) = \log[p(x; \xi)] \) and

\[
\mathbb{E}_{\xi}[f] = \int f(x)p(x; \xi)dx
\]

denotes the expectation with respect to the distribution \( p \).

The matrix \( G(\xi) \), called the Fisher information matrix (see \([6,7,8]\)), is symmetric and positive semi-definite. If \( G(\xi) \) is positive definite, then a Riemannian metric can be defined on the statistical manifold corresponding to the inner product induced on the natural basis of the coordinate system \( [\xi_i] \):

\[
g_{ij} = \langle \partial_i \mid \partial_j \rangle.
\]

This Riemannian metric is called the ”Fisher metric” or the ”information metric”. The corresponding geometric properties are characterized by the Christoffel symbols \( (\Gamma^i_{jk}) \) of the Levi-Civita connection with respect to the Fisher metric, which are defined solving :

\[
\sum_{j=1}^{2} g_{ih} \Gamma_{jk}^i = \Gamma_{jk,h}, \quad (h = 1, 2),
\]

where

\[
\Gamma_{jk,h}(\xi) = \mathbb{E}_{\xi}\left[\left( \partial_j \partial_h \ell_\xi + \frac{1}{2} \partial_j \partial_h \ell_\xi \right) \partial_k \ell_\xi \right],
\]

\( j, k = 1, 2, \ldots, n \).

The minimum distance between two elements \( f_1 \) and \( f_2 \) of a statistical manifold \( S \) is defined by the corresponding geodesic \( \omega \) which is the minimum length curve that connects them. Such a curve

\[
\omega = (\omega_t) : \mathbb{R} \rightarrow S
\]

satisfies the following system of 2\(\text{nd} \) order differential equations:

\[
\omega''(t) + \sum_{j,k=1}^{n} \Gamma_{jk}^i(t) \omega'(t) \omega'(t) = 0, \quad i = 1, 2, \ldots, n.
\]

under the conditions \( \omega(0) = f_1, \ \omega(1) = f_2 \).

In particular, it has been proved (see e.g. \([9,10]\)) that the two parameter Weibull distributions is a good choice for fitting wind and wave data. These distributions form a 2-dimensional statistical manifold with \( \xi = [\alpha, \beta], \ \Xi = \{[\alpha, \beta] \mid \alpha \beta > 0 \} \) (where \( \alpha \) is the shape and \( \beta \) the scale parameter) and

\[
p(x) = \frac{\alpha}{\beta} \left( \frac{x}{\beta} \right)^{\alpha-1} e^{-\left( \frac{x}{\beta} \right)^\alpha}, \quad \alpha, \beta > 0.
\]

The Fisher information matrix in this case becomes

\[
G(\alpha, \beta) = \begin{bmatrix}
\alpha(1-\gamma) & \beta(1-\gamma)
\end{bmatrix}
\]

where \( \gamma \) is the Euler gamma.

Let \( \xi_0 = [\alpha_0, \beta_0] \) and \( \xi_1 = [\alpha_1, \beta_1] \) be two members of the Weibull statistical manifold. Substituting the values of the Christoffel \( \Gamma^i_{jk} \) the previous system becomes:
In this section, we study the numerical solution of the BVP

\[
\begin{align*}
\omega''(t) + \frac{6(1 - \gamma)}{\pi^2 \beta_0} (\omega'(t))^2 + \frac{12(\gamma^2 - 2\gamma + \frac{2\gamma}{\omega^2} + 1)}{\pi^2 \alpha_0} \omega'_i(t) \omega'_j(t) - \frac{6(1 - \gamma)}{\pi^2 \beta_0} (\omega''(t))^2 &= 0 \\
\omega''(t) - \frac{\alpha_0^2}{\pi^2 \beta_0} (\omega'(t))^2 + \frac{12\alpha_0(1 - \gamma)}{\pi^2 \beta_0} \omega'_i(t) \omega'_j(t) - \frac{6(\gamma^2 - 2\gamma + \frac{2\gamma}{\omega^2} + 1)}{\pi^2 \alpha_0} (\omega''(t))^2 &= 0
\end{align*}
\]

with \( \omega(0) = \begin{bmatrix} \alpha_0 \\ \beta_0 \end{bmatrix} \), \( \omega(1) = \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} \) where

\( \omega(t) = \begin{bmatrix} \omega_1(t) \\ \omega_2(t) \end{bmatrix} \).

Our intention is to study the numerical solution of the above problem for a choice of its parameters obtained from environmental data sets within the South East Mediterranean territory.

It is worth noticing that IG techniques have been, directly or not, tested on different applications. Iguazúa and Olmo [11] utilized some of these ideas for geostatistical simulations for restricted samples. On the other hand, Cai et al. [12] applied information theoretic analysis on self-clustering of amino acids along protein chains. Finally, Resconi [13] based a risk analysis study on non-Euclidean geometric tools. However, up to the authors “knowledge”, applications of this framework on meteorology/oceanography is still very rare.

2.1 Finite Differences

We divide \([0,1]\) into \(N+1\) equal subintervals whose endpoints are at \(t_i = 0 + ih\), for \(i = 0, 1, \ldots, N, N+1\).

Assuming that the exact solution has a bounded fourth derivative we discretize and replace \(\omega_i(t), \omega'_i(t), \omega''_i(t)\) using the following standard finite differences (FD):

\[
\begin{align*}
\omega_i(t) &= \omega_0(i+1) - 2\omega_i(t) + \omega_0(t) - \frac{h^2}{12} \omega_0^{(4)}(\xi) \\
\omega'_i(t) &= \omega_0(i+1) - 2\omega_0(t) + \omega_0(t) - \frac{h^2}{12} \omega_0^{(4)}(\xi) \\
\omega''_i(t) &= \omega_0(i+1) - \omega_0(t-1) + \frac{h^2}{6} \omega_0^{(3)}(\eta) \\
\omega'_i(t) &= \omega_0(i+1) - \omega_0(t-1) + \frac{h^2}{6} \omega_0^{(3)}(\eta)
\end{align*}
\]

for some \(\xi, \eta\) in the interval \((t_{i-1}, t_{i+1})\).

The numerical difference method results from substituting the above FD to the differential equation, neglecting the error terms and employ the boundary conditions:

\[
\omega_0(0) = \omega_0(0) = \omega_0(1) = \omega_0^{N+1}, \omega_2(1) = \omega_2^{N+1},
\]

with \(\omega_0^i \approx \omega_0(t_i), \omega_2^i \approx \omega_2(t_i)\) for \(i = 1, \ldots, N\).

The result is a nonlinear system of \(2N\) equations with \(2N\) unknowns of the form

\[\hat{F}(\tilde{\omega}) = 0\]

where \(\theta = [0, \ldots, 0]^T\) and \(\tilde{\omega} = [\omega_0^1, \ldots, \omega_0^N, \omega_2^1, \ldots, \omega_2^N]^T\).

This system can be solved using the Newton’s iteration which is equivalent to solving in each step a linear system of the form:

\[J(\hat{\omega}^{(k-1)}) \cdot X = \hat{F}(\hat{\omega}^{(k-1)})\]

This can be done using the \(LU\) decomposition method and then update the solution

\[\hat{\omega}^{(k)} = \hat{\omega}^{(k-1)} - X.\]

2.2 The proposal of the \(LU\) modification

Amodio et al. [14] outlined the origin of almost block diagonal linear systems arising for solving boundary value ordinary differential equations including the case of finite differences approach. Their work refer to first order BVPs. Our problem is a system of second order equations which, when transformed to a system of order one differential equations, will double the dimension of the resulted the linear system . In our case, the resulted Jacobian matrix \(J\) has a specific special sparse form which we take advantage of when interchange its rows as follows.
I. Famelis et. al. : Classical and Quasi-Newton Methods for...

1. Famelis et. al. : Classical and Quasi-Newton Methods for...

75.0834

18

0.0176

] studied the cases that

]. Brezinski 

21

20

19

% of gain

0.3188

, 

. Brezinski

16

, Modified LU

45.6654

, 17

12

2

0.176

5

1

3

9

6

8

2

1

0.2686

Next Steps

In every step the number of elements which must be

1st Step: Make zero only two elements under the main
diagonal.

Update only 5 elements in rows 2 and 3 (the 2nd, 3rd,
(N + 1)-th, (N + 2)-th, (N + 3)-th).

Right here we have a significant reduction of floating
point operations as the classical LU updates the entries of
an (2N – 1) × (2N – 1) submatrix.

2nd Step: Make zero only 4 elements under the main
diagonal.

Update only 6 elements in rows 3 to 6 (the 3rd,4th,
(N + 1)-th, . . . , (N + 4)-th).

3rd Step: Make zero only 5 elements under the main
diagonal.

4rd-8th row: Update only 6 elements in rows 4 to 8 (the
5th,6th, (N + 1)-th, . . . , (N + 6)-th).

Next Steps Continue similarly until the upper
triangularization of J∗.

In every step the number of elements which must be
zeroed is increased by 1 until the N – 2-th step and the
number of elements in every row which must be updated
is increased by 1 until the (N-4)-th step. Then these
numbers are decreased by 1 in every step. The required
floating point operations for triangularizing the 2N × 2N
Jacobian matrix through modified LU factorization are

\( O \left( \frac{2N^3}{3} \right) \) whereas the classical one requires \( O \left( \frac{8N^3}{3} \right) \) flops. So, the classical Newton’s method demands \( O(k_0 \cdot \frac{8N^3}{3}) \) flops when using the classical LU factorization and \( O(k_0 \cdot \frac{2N^3}{3}) \) flops when applying the modified LU to the modified Jacobian matrix, for \( k_0 \) iterations.

The reduction is achieved in the first half of the
factorization, when we update specific entries and not
whole submatrices. The second half of the procedure
requires the same flops with classical LU. Eventually, the
modified LU is 4 times cheaper than the classical one, and
the execution time is expected to be even more fast, due
to the reduction of number of comparisons due to the zero
entries of the matrix \( J^* \).

In the next section, we compare the two LU
approaches for matrices which have the form of the
Jacobian \( J \) and random elements. We average the
computational time needed for sets 50 matrices. The
results are presented in Table 1.

### Table 1: LU vs modified LU factorization

<table>
<thead>
<tr>
<th>matrix dim.</th>
<th>classical LU</th>
<th>Modified LU</th>
<th>% of gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0.0176</td>
<td>0.0096</td>
<td>45.6654</td>
</tr>
<tr>
<td>500</td>
<td>0.3188</td>
<td>0.0753</td>
<td>76.3882</td>
</tr>
<tr>
<td>1000</td>
<td>3.8151</td>
<td>0.7283</td>
<td>80.9093</td>
</tr>
<tr>
<td>2000</td>
<td>35.2260</td>
<td>8.7771</td>
<td>75.0834</td>
</tr>
</tbody>
</table>

3 Quasi Newton methods

A variety of modifications of the classical Newton
method, substituting the Jacobian matrix with other
quantities, called Quasi Newton methods can be found in
literature [15,16,17,18,19]. Brezinski [20,21] classified
and proposed theoretically new Quasi-Newton methods.
We implemented numerically four of them. Since the
Jacobian matrix \( J^* \) of our system is of a special form, we
adapt these methods to \( J^* \) in order to reduce the required
floating point operations.

The general scheme of a Quasi Newton method reads

\[
\vec{\omega}^{(k)} = \vec{\omega}^{(k-1)} - \Lambda_{k-1} \cdot \tilde{F}(\vec{\omega}^{(k-1)}), \quad k = 0, 1, \ldots, \quad (5)
\]

where \( \Lambda_k \in \mathbb{R}^{2N \times 2N} \).

Brezinski [20] studied the cases that \( (\Lambda_k) \) is the identity
matrix multiplied by a scalar, \( (\Lambda_k) \) is a diagonal matrix
and \( (\Lambda_k) \) is a full matrix.

3.1 Scalar matrix case (SMC)

In this case \( \Lambda_k = \lambda_k \cdot I \), where

\[
\lambda_k = - \frac{\langle J(\vec{\omega}^{(k)}) F(\vec{\omega}^{(k)}), F(\vec{\omega}^{(k)}) \rangle}{\langle J(\vec{\omega}^{(k)}) F(\vec{\omega}^{(k)}), J(\vec{\omega}^{(k)}) F(\vec{\omega}^{(k)}) \rangle} \quad [20]
\]

and \( I \) the identity
Algorithm Scalar Matrix Case (SMC)

\[ f_d := J(\hat{\omega}^{(0)}) \cdot \hat{F}(\hat{\omega}^{(0)}) \]

\[ \lambda_0 = -\frac{(f_d, \hat{F}(\hat{\omega}^{(0)}))}{\|f_d\|^2} \]

\[ A_0 = \lambda_0 \cdot I \]

\[ \hat{\omega}^{(1)} = \hat{\omega}^{(0)} + A_0 \cdot \hat{F}(\hat{\omega}^{(0)}) \]

\[ k = 1 \]

\[ \text{while } \|\hat{\omega}^{(k)} - \hat{\omega}^{(k-1)}\| > TOL \text{ and } k < n_{max} \]

\[ f_d := J(\hat{\omega}^{(k-1)}) \cdot \hat{F}(\hat{\omega}^{(k-1)}) \]

\[ \lambda_{k-1} = -\frac{(f_d, \hat{F}(\hat{\omega}^{(k-1)}))}{\|f_d\|^2} \]

\[ A_{k-1} = \lambda_{k-1} \cdot I \]

\[ \hat{\omega}^{(k)} = \hat{\omega}^{(k-1)} + A_{k-1} \cdot \hat{F}(\hat{\omega}^{(k-1)}) \]

\[ \text{end while} \]

We use the modified LU factorization in order to compute \( J(\hat{\omega}^{(k-1)})^{-1} \cdot \hat{F}(\hat{\omega}^{(k-1)}) \) reducing significantly the required flops. The required flops are \( O(k_2 \cdot \frac{2N^2}{3}) \) for \( k_2 \) iterations plus the computation of \( \hat{F} \) and \( J \) at the point \( \hat{\omega}^{(k)} \) at every iteration. In general, the complexity for solving an \( 2N \times 2N \) system is of order \( O(k_2 \cdot \frac{2N^2}{3}) \).

3.3 Diagonal Matrix Case 2 (DMC2)

This is a modification of the previous case,

\[ \hat{\omega}^{(k)} = \hat{\omega}^{(k-1)} - \hat{F}(\hat{\omega}^{(k-1)}) \cdot \hat{\Lambda}_{k-1}, \]

where \( \hat{\Lambda}_{k} \) is computed using forward differences. Thus,

\[ \hat{\lambda}_k = A \hat{F}(\hat{\omega}^{(k-1)})^{-1} \cdot \Delta \hat{\omega}^{(k-1)}. \]

The DMC2 algorithm requires significant less flops than DMC1. More precisely, DMC2 demands \( O(k_2 \cdot 8N) \) flops for \( k_2 \) iterations plus the computation of \( \hat{F} \) and \( J \) at the point \( \hat{\omega}^{(k)} \) at every iteration.

3.4 Full matrix case (FMC)

Now, the matrix \( \Lambda_k \) is a full matrix with

\[ \Lambda_k = -\frac{\hat{F}(\hat{\omega}^{(k)}) \cdot \hat{F}(\hat{\omega}^{(k)})}{f_d} \]

where \( \hat{\omega}^{(k)} = \hat{\omega}^{(k)} + \lambda_0 \cdot \hat{F}(\hat{\omega}^{(0)}) \)

\[ \text{while } \|\hat{\omega}^{(k+1)} - \hat{\omega}^{(k)}\| > TOL \text{ and } k < n_{max} \]

\[ k = k + 1 \]

\[ f_d := J(\hat{\omega}^{(k)}) \cdot \hat{F}(\hat{\omega}^{(k)}) \]

\[ \lambda_{k-1} = diag(\hat{\lambda}_{k-1}) \]

\[ \hat{\omega}^{(k)} = \hat{\omega}^{(k-1)} + \lambda_{k-1} \cdot \hat{F}(\hat{\omega}^{(k-1)}) \]

\[ \text{end while} \]

The FMC algorithm demands \( O(k_3 \cdot 9N^2) \) flops at every iteration for \( k_3 \) iterations plus the computation of \( \hat{F} \) and \( J \) at the point \( \hat{\omega}^{(k)} \) at every iteration. In general, FMC requires \( O(k_3 \cdot (13N^2)) \) flops for solving an \( 2N \times 2N \) system of nonlinear equations.
Table 2: Levantine problems parameters

<table>
<thead>
<tr>
<th></th>
<th>model data no current</th>
<th>model data with current</th>
<th>satellite data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>shape α₀</td>
<td>scale β₀</td>
<td>shape α₁</td>
</tr>
<tr>
<td>Jan</td>
<td>1.600</td>
<td>1.010</td>
<td>1.726</td>
</tr>
<tr>
<td>Feb</td>
<td>1.500</td>
<td>1.400</td>
<td>1.571</td>
</tr>
<tr>
<td>Mar</td>
<td>1.402</td>
<td>1.132</td>
<td>1.571</td>
</tr>
<tr>
<td>Apr</td>
<td>1.564</td>
<td>0.695</td>
<td>1.719</td>
</tr>
<tr>
<td>May</td>
<td>1.533</td>
<td>0.608</td>
<td>1.608</td>
</tr>
<tr>
<td>Jun</td>
<td>2.333</td>
<td>0.633</td>
<td>2.582</td>
</tr>
<tr>
<td>Jul</td>
<td>2.557</td>
<td>0.837</td>
<td>2.688</td>
</tr>
<tr>
<td>Aug</td>
<td>3.099</td>
<td>0.716</td>
<td>3.341</td>
</tr>
<tr>
<td>Sep</td>
<td>2.418</td>
<td>0.754</td>
<td>2.580</td>
</tr>
<tr>
<td>Oct</td>
<td>1.629</td>
<td>0.551</td>
<td>1.850</td>
</tr>
<tr>
<td>Nov</td>
<td>1.846</td>
<td>0.392</td>
<td>1.959</td>
</tr>
<tr>
<td>Dec</td>
<td>1.435</td>
<td>1.216</td>
<td>1.512</td>
</tr>
</tbody>
</table>

4 Numerical Tests and Observations

For our numerical tests we choose data from the area of Levantine in the eastern Mediterranean Sea. For every month of year 2009 we have modeled wind speed and wave height data following two approaches. The former includes in the simulation the impact of sea currents, while the latter does not. Second source of data is the available corresponding satellite data. The data are fitted by a 2-parameter Weibull distribution and their corresponding shape and scale parameters are given in Table 2. When we consider the minimum length curve which connects the modeled and corresponding satellite data we conclude in 24 BV problems. For instance for the month January we have two problems, one for the curve connecting the modeled data with the presence of current with the satellite data, which we call “problem Jan with current”, and one for the modeled data without current to the satellite data, which we call “problem Jan with no current”.

To get reference solutions we use NDsolve of Mathematica to solve the 24 test problems. Mathematica uses the shooting method and one can set appropriate accuracy options (Working Precision, Accuracy Goal, Accuracy Goal) to get an considerably accurate solution which attains maximum value $1.11 \times 10^{-15}$. Moreover in Figure 3, the reference solution of the problem Aug with current is shown and in Figure 4, the defect for the reference solution of the same problem which attains a maximum value $6.25 \times 10^{-13}$. For this problem (along with a few other) the numerical integrator suggests elements of stiffness in the solution.

For our numerical tests we calculate and program the analytical form of $F$ and Jacobian $J$. We choose $N = 100$
and get a system of 200 equations. For the 24 problems we produce a reference solution and for an initial guess we use a perturbation of the initial data with random numbers. Then, we solve numerically the 24 test problems using the proposed methods for tolerances $10^{-3}, 10^{-4}, \ldots, 10^{-13}$ to compare efficiency and computational costs. We use two error measures at the grid points. The first one is $\|\hat{F}(\hat{\omega})\|_\infty$ the maximum absolute value that the numerical solution fails to satisfy the nonlinear problem resulting from the finite difference method. The second one is the $\|\hat{\omega}_0 - \hat{\omega}_{ref}\|_\infty$ maximum absolute value of the difference of the numerical solution and the reference solution. We investigate the sensitivity in the choice of the initial guess for tolerances $10^{-3}, 10^{-4}, \ldots, 10^{-13}$ with respect to its distance from the reference solution, in order to evaluate the range of convergence for each method.

We first solve the 24 problems using Newton’s Method with both the classical implementation of the LU factorization and the proposed modification and average the various measures. In Table 3, it is shown that both Newton’s method with classical LU and Newton’s method with modified LU have the same iterations and similar error measures at the grid points for all 24 problems. In some problems the iteration diverges for both approaches (e.g. problem Aug with current). The superiority of the proposed modified LU algorithm is obvious in Table 4, where we compare the average time to solve each of the 24 problems for 30 different choices of initial conditions using Newton’s method with classical LU and Newton’s method with modified LU.

### Table 3: Comparing the Classical LU and the proposed modified LU factorization.

<table>
<thead>
<tr>
<th>TOL</th>
<th>no of iter</th>
<th>time in secs</th>
<th>$|F(\hat{\omega})|_\infty$</th>
<th>$|\hat{\omega}<em>0 - \hat{\omega}</em>{ref}|_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-10}$</td>
<td>8.33</td>
<td>8.04</td>
<td>0.3529</td>
<td>0.0099</td>
</tr>
<tr>
<td>$10^{-15}$</td>
<td>8.7</td>
<td>8.67</td>
<td>0.1731</td>
<td>0.0119</td>
</tr>
</tbody>
</table>

In the following some remarks for the comparison of mod-NR, SMC, DMC1, DMC2, FMC are discussed. We solve for tolerances $10^{-3}, \ldots, 10^{-12}$ the 24 problems for a common initial condition each time and compare the average values of the results. It seems that DMC1 fails as the preconditioning matrix is singular. DMC2 succeeds only for $TOL = 10^{-3}, 10^{-4}$. For smaller tolerances it fails as $\Delta \hat{\omega} = \Delta \hat{F}(\hat{\omega})(k-1)^{-1}$. $\Delta \hat{\omega}$ the denominator becomes less than than unit round of (machine accuracy $\epsilon$). SMC, and FMC algorithms work for $TOL = 10^{-3}, \ldots, 10^{-8}$. For smaller tolerances, even if the methods do not seem to diverge, the iteration stops as the denominator of

$$
\# \frac{(J(\hat{\omega}(k)) \cdot \hat{F}(\hat{\omega}(k)), J(\hat{\omega}(k)) \cdot \hat{F}(\hat{\omega}(k)))}{(J(\hat{\omega}(k)) \cdot \hat{F}(\hat{\omega}(k)), J(\hat{\omega}(k)) \cdot \hat{F}(\hat{\omega}(k)))}
$$

becomes less than machine accuracy $\epsilon$.

In Tables 5 and 6, the modified Newton, SMC and FMC methods are compared for various tolerances in respect of the number of iterations and and the required running time. No mater the theoretical complexity cost, the time needed for the solution using Newton’s method is considerably smaller. SMC takes the longer time over the three methods. In Tables 7 and 8 the average
Table 7: Mean $\|\hat{F}(\hat{\omega}_{sol})\|_\infty$ for problems which converge.

<table>
<thead>
<tr>
<th>TOL</th>
<th>Newton’s method</th>
<th>SMC</th>
<th>FMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>1.55e-6</td>
<td>1.83e-3</td>
<td>9.92e-5</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>6.77e-8</td>
<td>1.45e-4</td>
<td>1.09e-3</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>7.29e-9</td>
<td>1.53e-5</td>
<td>1.13e-4</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>1.93e-9</td>
<td>1.48e-6</td>
<td>1.11e-5</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>8.24e-11</td>
<td>1.62e-7</td>
<td>1.19e-6</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>2.82e-11</td>
<td>1.71e-8</td>
<td>1.20e-7</td>
</tr>
</tbody>
</table>

Table 8: Mean $\|\hat{\omega}_{sol} - \hat{\omega}_{ref}\|_\infty$ for problems which converge.

<table>
<thead>
<tr>
<th>TOL</th>
<th>Newton’s method</th>
<th>SMC</th>
<th>FMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>2.87e-6</td>
<td>2.53e-1</td>
<td>4.17e-1</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>1.59e-5</td>
<td>3.77e-2</td>
<td>2.05e-1</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>3.49e-6</td>
<td>3.96e-3</td>
<td>2.96e-2</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>1.56e-5</td>
<td>3.94e-4</td>
<td>2.88e-3</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>2.53e-5</td>
<td>6.61e-5</td>
<td>3.29e-4</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>3.57e-6</td>
<td>7.14e-6</td>
<td>3.45e-5</td>
</tr>
</tbody>
</table>

Table 9: Newton’s method sensitivity in initial condition choice. Number of convergent solution of problems (out of 24).

<table>
<thead>
<tr>
<th>TOL</th>
<th>0.05</th>
<th>0.10</th>
<th>0.2</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>24</td>
<td>24</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>24</td>
<td>24</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>24</td>
<td>24</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>24</td>
<td>24</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>24</td>
<td>24</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>24</td>
<td>24</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>24</td>
<td>24</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>24</td>
<td>24</td>
<td>19</td>
<td>2</td>
</tr>
<tr>
<td>$10^{-11}$</td>
<td>24</td>
<td>24</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>$10^{-12}$</td>
<td>24</td>
<td>24</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>$10^{-13}$</td>
<td>24</td>
<td>22</td>
<td>19</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 10: SMC sensitivity in initial condition choice. Number of convergent solution of problems (out of 24).

<table>
<thead>
<tr>
<th>TOL</th>
<th>0.05</th>
<th>0.10</th>
<th>0.2</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 11: FMC sensitivity in initial condition choice. Number of convergent solution of problems (out of 24).

<table>
<thead>
<tr>
<th>TOL</th>
<th>0.05</th>
<th>0.10</th>
<th>0.2</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>15</td>
<td>14</td>
<td>10</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 12: SMC average $\|\hat{F}(\hat{\omega}_{sol})\|_\infty$ for all 24 problems.

<table>
<thead>
<tr>
<th>TOL</th>
<th>0.05</th>
<th>0.10</th>
<th>0.2</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>1.78e-3</td>
<td>1.61e-3</td>
<td>1.32e-3</td>
<td>9.40e-3</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>1.00e-4</td>
<td>1.00e-4</td>
<td>1.08e-4</td>
<td>1.11e-4</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>1.10e-5</td>
<td>1.10e-5</td>
<td>1.10e-5</td>
<td>1.11e-5</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>1.20e-6</td>
<td>1.19e-6</td>
<td>1.22e-6</td>
<td>1.21e-6</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>1.27e-7</td>
<td>1.07e-7</td>
<td>1.27e-6</td>
<td>1.26e-7</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>1.40e-8</td>
<td>1.40e-8</td>
<td>1.41e-8</td>
<td>1.34e-8</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>1.32e-8</td>
<td>1.20e-8</td>
<td>1.30e-8</td>
<td>1.25e-8</td>
</tr>
</tbody>
</table>
developed within the framework of Information Geometry. In this way, we adopt an approach that avoids simplifications connected with the a priori acceptance of Euclidean distances for the data under study as in the least square based methods. More precisely, Weibull probability distribution functions are fitted to the data under study and treated as elements of statistical manifolds on which Riemannian metrics are defined and the distances between the different sets of data are measured by means of the corresponding geodesics (i.e. minimum length curves).

These curves are obtained as numerical solutions of second order boundary value problems using finite differences. For the numerical solution of the resulting system of nonlinear equations, we apply Newton and Quasi Newton methods in which we take into account the special form of the Jacobian matrix and modify appropriately the LU algorithm in order to reduce the computational complexity.

More precisely, the Newton method using the modified LU factorization demands the same iterations and has an absolute error of the same order with the classical one but it requires significantly less floating point operations. The error measures for Newton’s method (classical or modified) are considerably smaller and the method converges for smaller tolerances compared to the SMC and FMC.

The DMC methods are not proposed for our problem, since, the preconditioning matrix is almost singular.

The Newton’s method using either classical or modified LU factorization is more sensitive in the selection of the initial point compared to SMC and FMC since, for some problems, it demands the initial values to be closer to the final solution than the other two methods.

The use of SMC or FMC as an initial procedure improves the behavior of Newton’s method in the cases that diverges. These methods can be used as starting procedures that compute initial points for the Newton’s iteration within the interval of convergence, while the modified LU factorization reduces significantly the required floating point operations resulting to an effective algorithm for solving efficiently the system of non linear equations.

### Acknowledgement

The authors would like to thank the “The Atmospheric Modeling and Weather Forecasting Group” of the Department of Physics, University of Athens, Greece for providing the meteorological data.

This work has been co-funded by the European Union and Greek national resources under the framework of the “Archimedes III: Funding of Research Groups in TEI of Athens” project of the “Education and Lifelong Learning” Operational Programme.

---

**Table 13:** FMC average $\|F(\mathbf{w}_{old})\|_\infty$ for all 24 problems.

<table>
<thead>
<tr>
<th>TOL</th>
<th>0.05</th>
<th>0.10</th>
<th>0.2</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>7.14e-3</td>
<td>7.80e-3</td>
<td>8.16e-3</td>
<td>8.27e-3</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>8.46e-4</td>
<td>8.31e-4</td>
<td>8.46e-4</td>
<td>8.78e-4</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>1.02e-5</td>
<td>1.03e-5</td>
<td>1.04e-5</td>
<td>1.04e-5</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>1.08e-6</td>
<td>1.07e-6</td>
<td>1.09e-6</td>
<td>1.06e-6</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>1.09e-7</td>
<td>1.11e-7</td>
<td>1.15e-7</td>
<td>1.15e-7</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>1.30e-8</td>
<td>1.34e-8</td>
<td>1.42e-8</td>
<td>1.31e-8</td>
</tr>
</tbody>
</table>

**Table 14:** SMC and FMC as starting procedures.

<table>
<thead>
<tr>
<th>TOL $= 10^{-11}$</th>
<th>no of iter.</th>
<th>$|F(\mathbf{w}<em>{old})|</em>\infty$</th>
<th>$|\mathbf{w}<em>{old} - \mathbf{w}</em>{ref}|_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton’s methodR</td>
<td>182</td>
<td>4.38e+17</td>
<td>9.03e+88</td>
</tr>
<tr>
<td>SMC</td>
<td>9219</td>
<td>1.75e-6</td>
<td>7.51e-4</td>
</tr>
<tr>
<td>Newton’s methodR</td>
<td>7</td>
<td>4.78e-14</td>
<td>3.74e-4</td>
</tr>
<tr>
<td>FMC</td>
<td>5602</td>
<td>1.39e-5</td>
<td>3.40e-3</td>
</tr>
<tr>
<td>Newton’s methodR</td>
<td>7</td>
<td>5.59e-14</td>
<td>3.74e-4</td>
</tr>
</tbody>
</table>

An interesting remark is that Newton’s method with either classical and modified LU factorization seem to be very sensitive in the choice of initial guess and so they have shorter interval of convergence compare to both SMC, FMC. This can be tested by solving the problems taking initial guesses further away from the reference solution (see Tables 9, 10, 11). It must be noted that SMC and FMC do not diverge for TOL $= 10^{-9}$ and smaller (see Tables 12, 13). The algorithms stop as the denominator of (6) becomes less than machine accuracy $\varepsilon$. Even though, as Newton’s method is more accurate SMC and FMC can be used as starting procedures. For problems such as Aug with current, where Newtons method diverges, we can use either SMC or FMC to get an initial point and then continue with Newton’s iteration (see Table 14).

**5 Conclusions**

In this paper we study the numerical solution of a special boundary value problem, arising in meteorological parameters prediction. The construction and study of the cost functions that estimate the discrepancies between modeled and observed data is based on techniques recorded error measures for the modified Newton, SMC and FMC methods are presented. Newtons method attains a better convergence (with respect to the reference solution) and has a smaller defect for the nonlinear system compared to its competitors.

An interesting remark is that Newton’s method with either classical and modified LU factorization seem to be very sensitive in the choice of initial guess and so they have shorter interval of convergence compare to both SMC, FMC. This can be tested by solving the problems taking initial guesses further away from the reference solution (see Tables 9, 10, 11). It must be noted that SMC and FMC do not diverge for TOL $= 10^{-9}$ and smaller (see Tables 12, 13). The algorithms stop as the denominator of (6) becomes less than machine accuracy $\varepsilon$. Even though, as Newton’s method is more accurate SMC and FMC can be used as starting procedures. For problems such as Aug with current, where Newtons method diverges, we can use either SMC or FMC to get an initial point and then continue with Newton’s iteration (see Table 14).
References

Dimitrios Triantafyllou
received the PhD degree in Mathematics at National Kapodistrian University of Athens. He has worked as contracted Lecturer at Athens University of Economics and Business, Hellenic Army Academy, Hellenic Naval Academy and Technological Educational Institute of Athens. His research interests are in the areas of Computational Mathematics, Numerical Analysis, and Numerical Linear Algebra and more specific in factorization of special structured matrices such as Sylvester and Generalised Sylvester matrices, Block Toeplitz matrices etc., computation of Rank, nullity and null space of matrices, Numerical, Symbolical and Hybrid methods for the computation of the Greatest Common Divisor (GCD) of polynomials. He is referee of mathematical journals.