A study of QR decomposition and Kalman filter implementations

DAVID FUERTES RONCERO

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden September 2014

XR-EE-SB 2014:010

KTH Electrical Engineering

A study of QR decomposition and Kalman filter implementations

DAVID FUERTES RONCERO

Stockholm 2014

Signal Processing
School of Electrical Engineering
Kungliga Tekniska Hgskolan

XR-EE-SB 2014:010

Abstract

With the rapid development of new technologies during the last decades, the demand of complex algorithms to work in real-time applications has increased considerably. To achieve the real time expectations and to assure the stability and accuracy of the systems, the application of numerical methods and matrix decompositions have been studied as a trade-off between complexity, stability and accuracy. In the first part of this thesis, a survey of state-of-the-art QR Decomposition methods applied to matrix inversion is done. Stability and accuracy of these methods are analyzed analytically and the complexity is studied in terms of operations and level of parallelism. Besides, a new method called Modified Gaussian Elimination (MGE) is proposed. This method is shown to have better accuracy and less complexity than the previous methods while keeping good stability in real time applications. In the second part of this thesis, different techniques of extended Kalman Filter implementations are discussed. The EKF is known to be numerically unstable and various methods have been proposed in the literature to improve the performance of the filter. These methods include square-root and unscented versions of the filter that make use of numerical methods such as QR, LDL and Cholesky Decomposition. At the end of the analysis, the audience/reader will get some idea about best implementation of the filter given some specifications.

Acknowledgment

The completion of this thesis would not have been possible without the participation and support of many people, so I would like to dedicate some words to thank all the people that has make this thesis possible.

First of all, I would like to thank my supervisor Satyam Dwivedi for giving me the opportunity of working in that project with him and helping me in the hard moments. I have enjoyed all the moments of my research, and I really appreciate the opportunity to study state-of-the-art publications that make an actual impact in the real world. My work in this thesis have also encourage me to focus my career in research.

I need also to thank my parents, my sister and the rest of my family for their unconditional support. Without their support during all the years of my life, I would not have become the person that I am today, for which I will be eternally grateful.

Finally, but not less important, I want to thank all my friends and all the people that I have known here and have make me feel like home. Specially, I want to thank to Pedro, Dani, Julia, Cesc, Pablo, Gonzalo and Carles for all the good and bad moments that we have lived together during our five years at university. Thanks to you, this is a period of my life that I will never forget. I can not finish without saying thanks to Xavi, Albert, Adrian, Aida, Marcos, Sergi and Marina for being my friends. I know that no matter the distance, you will always be there if I need you.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

With the rapid development of new technologies during the last decades, the demand of complex algorithms to work in real-time applications has increased considerably. To achieve the real time expectations and to assure the stability and accuracy of the systems, the application of numerical methods and matrix decompositions have been studied as a trade-off between complexity, stability and accuracy.

Matrix inversion is one the most important operations in signal processing and is present in many areas such as positioning systems, wireless communications or image processing. The computation of a matrix inverse grows in exponent with the matrix dimension and can introduce errors and instabilities in the system. For this reason, it is worthy to study all the existing methods to accelerate the computation and assure the stability of the operation. The methods developed during the last decades rely on matrix decompositions to reduce complexity. These decompositions include SVD, QR Decomposition and LU Decomposition. QR Decomposition is the most widely used since assures stability and have less complexity. In the first part of the thesis, a survey of the state-of-the-art QR Decomposition methods applied to matrix inversion is done. The stability and accuracy of these methods are analysed analytically and the complexity is studied in terms of operations and level of parallelism. Besides, a new method called Modified Gaussian Elimination (MGE) is proposed. This method is shown to have better accuracy and less complexity than the previous methods while keeping good stability in real applications.

In the second part of the thesis, different techniques of Extended Kalman Filter implementations are discussed. The EKF is known to be numerically

unstable and various methods have been proposed in the literature to improve the performance of the filter. These methods include square-root and unscented versions of the filter that make use of numerical methods such as QR, LDL and Cholesky Decomposition. Different implementations of these methods applied to EKF have been studied for trade-off between complexity and accuracy. At the end of the analysis, the audience/reader will get some idea about best implementation of the filter given some specifications.

Chapter 2

QR Decomposition

First of all, lets introduce QR Decomposition. In QR decomposition, an $\mathbf{A}$ matrix of size $m \times n$ is written as the product of an upper-triangular matrix $\mathbf{R}$ of size $m \times n$ and an orthogonal matrix $\mathbf{Q}$ of size $m \times m$

$$\mathbf{A} = \mathbf{QR}. \tag{2.1}$$

After decomposing matrix $\mathbf{A}$ in the $\mathbf{QR}$ form, $\mathbf{A^{-1}}$ can be easily computed as

$$\mathbf{A^{-1}} = \mathbf{R^{-1}Q^{-1}} = \mathbf{R^{-1}Q^{H}}, \tag{2.2}$$

where $\mathbf{R^{-1}}$ can be computed using backward substitution (2.3), leading to a much more faster and efficient computation than the direct inversion of matrix $\mathbf{A}$. The algorithm is given in Alg. 1.

$$r_{j,i}^{-1} = \frac{-\sum_{k=j}^{i-1} r_{j,k+1} r_{k+1,i}^{-1}}{r_{j,j}}. \tag{2.3}$$

The most common methods to obtain QR are Householder Reflectors (HR) [2], Modified Gram-Schmidt (MGS) [3] and Givens Rotations (GR) [2]. CORDIC functions can also be used to obtain the QR Decomposition using vectoring and rotation mode to perform angle calculations and rotation operations directly [4]. To reduce complexity and improve the numerical properties of the transformation, new methods such as Modified Squared Givens Rotations (MSGR) [5] have been proposed recently to avoid the use of costly operations such as square roots. This method was developed to compute the QR Decomposition and matrix inverse of complex matrices, but can also be applied in the real case.

This chapter is organized as follows. In Section 2.1 all the mentioned methods, as well as MGE, are introduced, the algorithms are given and the

Algorithm 1: Backward Substitution

```
1  R' = 0_{n×n}
2  for i = n to 1 do
3  |  R'_{i,i} = 1/R_{i,i}
4  |  for j = i - 1 to 1 do
5  |  |  for k = j to i-1 do
6  |  |  |  R'_{j,i} = R'_{j,i} − R_{j,k+1} R'_{k+1,i}
7  |  |  end
8  |  |  R'_{j,i} = R'_{j,i} / R_{j,j}
9  |  end
10 end
```

operations that can introduce errors and instabilities are discussed. In Section 2.2 the complexity of the methods in terms of number of operations is studied, numerical properties are studied in Section 2.3, an efficient implementation of MGE is proposed in 2.6 and finally the conclusions are given in Section 2.7.

2.1 Algorithms

In this section, the methods for computing QR Decomposition are briefly described, the algorithms are given and the operations involve are discuss in detail with a remark in the operations that can cause over/under-flows and introduce large errors.

2.1.1 Householder Reflectors (HR)

Householder's method is based on the idea of multiplying matrix $\mathbf{A}$ by a set of orthogonal matrices such that every matrix put zeros below the diagonal of each column. The method is detailed in [2]. Doing that we obtain matrix $\mathbf{R}$ as

$$\begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} \overset{\mathbf{Q_1}}{\rightarrow} \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \overset{\mathbf{Q_2}}{\rightarrow} \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} \overset{\mathbf{Q_3}}{\rightarrow} \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{pmatrix}. \tag{2.4}$$

To obtain matrix $\mathbf{Q}$ we just need to multiply and transpose all the orthogonal matrices

$$\mathbf{Q_1 Q_2 ... Q_n} = \mathbf{Q^{-1}} = \mathbf{Q^H}. \tag{2.5}$$

The computation of $\mathbf{Q^{-1}}$ is very straightforward if we see the transformation as a product of matrices

$$\mathbf{Q_1 Q_2 ... Q_n A} = \mathbf{R}, \tag{2.6}$$

so

$$\mathbf{Q^{-1}} = \mathbf{Q_1 Q_2 ... Q_n}. \tag{2.7}$$

Then $\mathbf{Q^{-1}}$ is obtain applying the same transformations to the Identity Matrix than to matrix $\mathbf{A}$

$$\mathbf{Q^{-1} I} = \mathbf{Q^{-1}} = \mathbf{Q^H}. \tag{2.8}$$

The algorithm is given in Alg. 2. Looking at the operations, a large number of multiplications is needed to compute the norm operations in lines 5 and 7 and the vector operation in line 8. These operations not only increase the complexity of the algorithm, they also generate very large numbers that increase the dynamic range of the algorithm, incrementing the quantization errors and leading to overflows if the input is not properly normalized. Besides, the square root in line 6 and the division in line 9 are complex to compute in processors.

Algorithm 2: Householder Reflectors

1 $\mathbf{R} = \mathbf{A}$
2 $\mathbf{Q} = \mathbf{I}_{n \times n}$
3 **for** $k = 1$ ***to*** n **do**
4 $\quad \mathbf{v} = \mathbf{R}_{k:n,k}$
5 $\quad a = ||\mathbf{v}||_2^2$
6 $\quad v_1 = v_1 + sign(v_1)\sqrt{a}$
7 $\quad a = ||\mathbf{v}||_2^2$
8 $\quad \mathbf{B} = \mathbf{v}\mathbf{v}'$
9 $\quad \mathbf{F} = \mathbf{I} - \frac{2}{a}\mathbf{B}$
10 $\quad \mathbf{R}_{k:n,k:n} = \mathbf{F}\mathbf{R}_{k:n,k:n}$
11 $\quad \mathbf{Q}_{k:n,1:n} = \mathbf{F}\mathbf{Q}_{k:n,1:n}$
12 **end**

2.1.2 Modified Gram-Schmidt (MGS)

The Gram-Schmidt process is a method for orthonormalising a set of vectors, in our case the columns of matrix $\mathbf{A}$. The process finds orthonormal vectors

that span the successive spaces spanned by the columns of $\mathbf{A}$. That means that the first column of $\mathbf{A}$ ($\mathbf{a_1}$) only depends on one vector ($\mathbf{q_1}$), the second column of $\mathbf{A}$ ($\mathbf{a_2}$) only depends on the previous vector and a new orthogonal vector ($\mathbf{q_2}$), the third column of A ($\mathbf{a_3}$) only depends on the two previous orthogonal vectors and a new vector orthogonal to the previous ones ($\mathbf{q_3}$) and so on. That leads directly to the QR decomposition

$$\begin{pmatrix} \mathbf{a_1} & \mathbf{a_2} & \mathbf{a_3} & \cdots & \mathbf{a_n} \end{pmatrix} = \begin{pmatrix} \mathbf{q_1} & \mathbf{q_2} & \mathbf{q_3} & \cdots & \mathbf{q_n} \end{pmatrix} \begin{pmatrix} \mathbf{r_{11}} & \mathbf{r_{12}} & \mathbf{r_{13}} & \cdots & \mathbf{r_{1n}} \\ & \mathbf{r_{22}} & \mathbf{r_{23}} & \cdots & \mathbf{r_{2n}} \\ & & \mathbf{r_{33}} & \cdots & \mathbf{r_{2n}} \\ & & & \ddots & \vdots \\ & & & & \mathbf{r_{nn}} \end{pmatrix}. \tag{2.9}$$

The classical Gram-Schmidt process is known to be numerically unstable and very sensible to rounding errors, which cause a lose of orthogonality in the vectors $\mathbf{q_j}$. However, a different approach called Modified Gram-Schmidt using the same idea is less sensible to rounding errors and therefore have better numerical properties [3].

The algorithm is given in Alg. 3. In the case of MGS, the most critical operations are the multiplications in lines 6 and 7 and the norm computation in line 9. Divisions and square roots are also computed in lines 9 and 10.

Algorithm 3: Modified Gram-Schmidt

1 $\mathbf{R} = \mathbf{0}_{n\times n}$
2 $\mathbf{Q} = \mathbf{I}_{n\times n}$
3 **for** *j = 1* ***to*** *n* **do**
4 $\quad \mathbf{w}_{1:n,j} = \mathbf{A}_{1:n,j}$
5 $\quad$ **for** *i = 1* ***to*** *j-1* **do**
6 $\quad\quad \mathbf{R}_{i,j} = \mathbf{w}'_{1:n,j}\mathbf{Q}_{1:n,i}$
7 $\quad\quad \mathbf{w}_{1:n,j} = \mathbf{w}_{1:n,j} - \mathbf{R}_{i,j}\mathbf{Q}_{1:n,i}$
8 $\quad$ **end**
9 $\quad \mathbf{R}_{j,j} = \sqrt{\mathbf{w}'_{1:n,j}\mathbf{w}_{1:n,j}}$
10 $\quad \mathbf{Q}_{1:n,j} = \frac{\mathbf{w}_{1:n,j}}{r_{j,j}}$
11 **end**

2.1.3 Givens Rotations (GR)

The method introduced by Givens rotates the elements of $\mathbf{A}$ in pairs such that one of the elements is set to 0 [2]. After applying rotations to all the elements of the lower triangular part of $\mathbf{A}$, we obtain matrix $\mathbf{R}$. To obtain

matrix $\mathbf{Q}$ we follow the same method as in HR, applying the same transformations to the Identity Matrix than to matrix $\mathbf{A}$. A sequence of the transformation for a 4×4 matrix is

$$\begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} \overset{\mathbf{Q}_1}{\rightarrow} \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \end{pmatrix} \overset{\mathbf{Q}_2}{\rightarrow} \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \overset{\mathbf{Q}_3}{\rightarrow} \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} \overset{\mathbf{Q}_4}{\rightarrow}$$
$$\begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix} \overset{\mathbf{Q}_5}{\rightarrow} \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} \overset{\mathbf{Q}_6}{\rightarrow} \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{pmatrix}. \tag{2.10}$$

The rotation is performed using a 2×2 matrix of the form

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}. \tag{2.11}$$

This matrix is orthogonal ($|c|^2 + |s|^2 = 1$) and the elements c and s are the sine and cosine of the transformation. To rotate a vector $\begin{pmatrix} x & y \end{pmatrix}$ into $\begin{pmatrix} x' & 0 \end{pmatrix}$, the values of c and s are

$$c = \frac{x}{\sqrt{x^2 + y^2}}; s = \frac{-y}{\sqrt{x^2 + y^2}}. \tag{2.12}$$

The algorithm is given in Alg. 4. The products and sum in line 5 generate large numbers and can affect the accuracy of the decomposition. Divisions and square roots are computed in lines 5, 6 and 7. The number of iterations of the algorithm is higher in the case of GR and therefore more square roots and divisions are computed than in the case of HR and MGS. The rotation operation in lines 8 and 9 can also create problems of over/under-flows in the computation of the algorithm if the dynamic range is not well adjusted.

2.1.4 CORDIC Givens Rotations(CORDIC)

As mentioned above, multiplications, divisions and square roots are very costly operations and are a limitation in the computation time of all the previous methods. To avoid the use of these operations, an algorithm called CORDIC was developed in 1959 by Jack Volder [6]. The advantage of CORDIC functions is that compute trigonometric and linear functions using only shifts and additions. On the other hand, CORDIC functions are iterative operations and can increase the latency of the system. Besides, the area cost of implementing CORDIC is very high.

CORDIC can be use in two different modes: rotation mode and vectoring mode. The rotation mode can be use to rotate a vector given an angle and

Algorithm 4: Givens Rotations

$\mathbf{R} = \mathbf{A}$
$\mathbf{Q} = \mathbf{I}_{n\times n}$
for $i = j$ ***to*** n **do**
  for $i = m$ - 1 ***to*** $j + 1$ **do**
    $b = \sqrt{r_{i-1,j}^2 + r_{i,j}^2}$
    $a_1 = \frac{r_{i-1,j}}{b}$
    $a_2 = -\frac{r_{i,j}}{b}$
    $\mathbf{R}_{i-1:i,j:n} = [a_1, -a_2; a_2, a_1]\mathbf{R}_{i-1:i,j:n}$
    $\mathbf{Q}_{i-1:i,1:n} = [a_1, -a_2; a_2, a_1]\mathbf{Q}_{i-1:i,1:n}$
  end
end

the vectoring mode can be use to calculate the angle need for the rotation. Therefore, the CORDIC based Givens Rotations first uses the vectoring mode to calculate the angle of rotation and after that uses the rotation mode to rotate the two rows involve in the rotation. The algorithm is given in Alg. 5 and the rotation and vectoring mode are detailed in [4].

2.1.5 Modified Squared Givens Rotations (MSGR)

An improved version of Givens Rotations was presented in [5]. This algorithm is based in translating the rows to rotate to an U and V-space such that the rotation of the matrices can be done with very simple operations. The equations to translate the rows to the new space and to do the rotation are shown in (2.13) and (2.14) respectively, where $\mathbf{r}$ and $\mathbf{a}$ are the rows to rotate and $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ are the rotated rows. The constant w_a can be set to 1 to simplify the operations without significant loss of accuracy.

$$\begin{aligned} \mathbf{u} &= r_1^*\mathbf{r}, \\ \mathbf{a} &= w_a^{\frac{1}{2}}\mathbf{v}, \end{aligned} \tag{2.13}$$

$$\begin{aligned} \bar{\mathbf{u}} &= \mathbf{u} + wv_k^*\mathbf{v}, \\ \bar{\mathbf{v}} &= \mathbf{v} - (\tfrac{v_k}{u_k})\mathbf{u}. \end{aligned} \tag{2.14}$$

This new algorithm requires less operations than Givens Rotations and avoid the use of square roots in the computation. In the paper is shown how using this algorithm it is possible to achieve very low latency and high performance. Besides, stability issues when processing zero values are solved using partial pivoting of rows when the element to process is zero. This algorithm

Algorithm 5: CORDIC Givens Rotations

```
1  R = A
2  Q = I_{n×n}
3  for i = j to n do
4      for i = m - 1 to j + 1 do
5          CORDIC Vectoring Mode to calculate angle of rotation
6           Input Vector:   (R_{i-1,j}, R_{i,j})
            Output angle:   θ
7          for k = 1 to n do
8              CORDIC Rotation Mode to rotate the matrix (x2)
                Input angle:          θ
                Input Vector 1:       (r_{i-1,k}, r_{i,k})
9               Output Vector 1:      (r_{i-1,k}, r_{i,k})
                Input Vector 2:       (q_{i-1,k}, q_{i,k})
                Output Vector 2:      (q_{i-1,k}, q_{i,k})
10         end
11     end
12 end
```

does not perform a pure QR decomposition since the resulting matrix $\mathbf{Q}$ is not orthogonal. However, the inverse of matrix $\mathbf{Q}$ can be computed directly from the algorithm (2.8), so the algorithm is still useful to compute matrix inverses.

The algorithm is given in Alg. 6. In the case of MSGR, the most critical operations that can lead to instabilities and loss of accuracy are the translation to the U space in lines 4 and 5 and the update in lines 22 and 24 for the same reason than in the previous algorithms.

2.1.6 Modified Gaussian Elimination (MGE)

The method to obtain the QR Decomposition that we propose here is based on the idea of computing the triangular matrix $\mathbf{R}$ applying Gaussian Elimination. Considering a $n \times n$ matrix

Algorithm 6: MSGR

$\mathbf{R} = \mathbf{A}$
$\mathbf{Q} = \mathbf{I}_{n \times n}$
for $j = 1$ ***to*** n **do**
  $\mathbf{u}_1 = r_{j,j}\mathbf{R}_{j,j:n}$
  $\mathbf{u}_2 = r_{j,j}\mathbf{Q}_{j,1:n}$
  for $i = j + 1$ ***to*** m **do**
    $\mathbf{v}_1 = \mathbf{R}_{i,j:n}$
    $\mathbf{v}_2 = \mathbf{Q}_{1,1:m}$
    if $u_1(1)$ ***equal to*** 0 **then**
      if $v_1(1)$ ***different to*** 0 **then**
        $\mathbf{R}_{i,j:n} = -\mathbf{R}_{j,j:n}$
        $\mathbf{R}_{j,j:n} = v_1(1)'\mathbf{v}_1$
        $\mathbf{Q}_{i,1:n} = -\mathbf{Q}_{i,1:n}$
        $\mathbf{Q}_{j,1:n} = v_1(1)'\mathbf{v}_2$
      else
        $\mathbf{R}_{i,j:n} = -\mathbf{R}_{j,j:n}$
        $\mathbf{R}_{j,j:n} = \mathbf{v}_1$
        $\mathbf{Q}_{i,1:n} = -\mathbf{Q}_{i,1:n}$
        $\mathbf{Q}_{j,1:n} = \mathbf{v}_2$
      end
    else
      $\mathbf{R}_{j,j:n} = \mathbf{u}_1 + v_1(1)\mathbf{v}_1$
      $\mathbf{R}_{i,j:n} = \mathbf{v}_1 - \frac{v_1(1)}{u_1(1)}\mathbf{u}_1$
      $\mathbf{Q}_{j,1:n} = \mathbf{u}_2 + v_1(1)\mathbf{v}_2$
      $\mathbf{Q}_{i,1:n} = \mathbf{v}_2 - \frac{v(1)_1}{u_1(1)}\mathbf{u}_2$
    end
  end
end

$$\mathbf{A} = \begin{pmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \vdots \\ \mathbf{a_i} \\ \vdots \\ \mathbf{a_n} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn} \end{pmatrix}, \tag{2.15}$$

to zero out the element in row i and column j we apply the Gaussian Elimination equation

$$\mathbf{a_i} = \mathbf{a_i} - \frac{a_{i,j}}{aj,j}\mathbf{a_j}. \tag{2.16}$$

After applying the transformation we obtain the triangular matrix $\mathbf{R}$. As in MSGR, matrix $\mathbf{Q}$ is not orthogonal and $\mathbf{Q^{-1}}$ is directly obtain from the algorithm applying the rotations to the Identity Matrix.

To illustrate how the algorithm works, an example is presented here. Lets consider the following 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} 1 & -2 & -6 \\ 2 & 4 & 12 \\ 1 & -4 & -14 \end{pmatrix}. \tag{2.17}$$

The intermediate steps to triangularize $\mathbf{A}$ applying (2.16) are

$$\begin{pmatrix} 1 & -2 & -6 \\ 2 & 4 & 12 \\ 1 & -4 & -14 \end{pmatrix} \overset{\mathbf{Q_1}}{\rightarrow} \begin{pmatrix} 1 & -2 & -6 \\ 0 & 8 & 24 \\ 1 & -4 & -14 \end{pmatrix} \overset{\mathbf{Q_2}}{\rightarrow} \begin{pmatrix} 1 & -2 & -6 \\ 0 & 8 & 24 \\ 0 & -2 & -8 \end{pmatrix} \overset{\mathbf{Q_3}}{\rightarrow} \begin{pmatrix} 1 & -2 & -6 \\ 0 & 8 & 24 \\ 0 & 0 & -2 \end{pmatrix} = \mathbf{R}. \tag{2.18}$$

As mentioned above, to obtain $\mathbf{Q^{-1}}$ we apply the same transformations to the Identity Matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \overset{\mathbf{Q_1}}{\rightarrow} \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \overset{\mathbf{Q_2}}{\rightarrow} \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \overset{\mathbf{Q_3}}{\rightarrow} \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0.25 & 1 \end{pmatrix} = \mathbf{Q^{-1}}. \tag{2.19}$$

As can be seen in the example, the resulting matrix $\mathbf{Q^{-1}}$ is lower-triangular, so the method is similar to an LU Decomposition. The difference is that in this method the algorithm gives us directly the inverse of matrix $\mathbf{L}$ and to compute the inverse we just need to compute $\mathbf{R^{-1}}$ using backward substitution and multiply both matrices. This procedure have the advantage that is less complex than previous LU methods to compute inverses and can also be implemented efficiently in parallel structures. To avoid stability issues when

processing zero values, a partial pivoting of rows when the element to process is zero is also done.

The Modified Gaussian Elimination algorithm is given in Alg. 7. The MGE algorithm is very similar to MSGR but avoiding the computation of the operations that can produce problems in the system. For that reason, the algorithm have a smaller dynamic range and is robust against over/underflows.

Algorithm 7: Modified Gaussian Elimination

1 $\mathbf{R} = \mathbf{A}$
2 $\mathbf{Q} = \mathbf{I}_{n \times n}$
3 **for** $j = 1$ ***to*** n **do**
4 **for** $i = j{+}1$ ***to*** n **do**
5 **if** $\boldsymbol{R}_{j,j}$ ***equal to*** 0 **then**
6 $\mathbf{v}_1 = \mathbf{R}_{i,j:n}$
7 $\mathbf{R}_{i,j:n} = \mathbf{R}_{j,j:n}$
8 $\mathbf{R}_{j,j:n} = \mathbf{v}_1$
9 $\mathbf{v}_2 = \mathbf{Q}_{i,1:n}$
10 $\mathbf{Q}_{i,1:n} = \mathbf{Q}_{j,1:n}$
11 $\mathbf{Q}_{j,1:n} = \mathbf{v}_2$
12 **end**
13 $\mathbf{Q}_{i,1:n} = \mathbf{Q}_{i,1:n} - \frac{r_{i,j}}{r_{j,j}} \mathbf{Q}_{j,1:n}$
14 $\mathbf{R}_{i,j:n} = \mathbf{R}_{i,j:n} - \frac{r_{i,j}}{r_{j,j}} \mathbf{R}_{j,j:n}$
15 **end**
16 **end**

2.2 Computation complexity

In this section, the complexity of the different methods is compared in terms of number of operations. A remark about the level of parallelism of the algorithms is also done. The number of operations of HR, MGS, GR and MSGR can be found in the literature. To compute the number of operations of MGE we consider the worse case when no pivoting is done. Looking at Alg. 7, we can see that the operations of lines 13 and 14 compute $n + j$ multiplications, $n + j$ subtractions and 1 division every iteration. So, the

Method	Asymptotic cost
HR [2]	$\sim \frac{4n^3}{3}$
MGS [2]	$\sim 2n^3$
GR [2]	$\sim 5n^3$
CORDIC [4]	$\sim 5n^3p$
MSGR [5]	$\sim \frac{15n^3}{6}$
MGE	$\sim \frac{10n^3}{6}$

Table 2.1: Asymptotic number of operations for QR based matrix inverse ($n \times n$ matrix)

total number of operations computed by the algorithm is

$$\begin{aligned} Subtractions &: \sum_{j=1}^{n} \sum_{i=1}^{j-1} n + j = \frac{5n^3}{6} - \frac{n^2}{2} - \frac{n}{3} \\ Multiplications &: \sum_{j=1}^{n} \sum_{i=1}^{j-1} n + j = \frac{5n^3}{6} - \frac{n^2}{2} - \frac{n}{3} \\ Divisions &: \sum_{j=1}^{n} \sum_{i=1}^{j-1} 1 = \frac{n^2}{2} - \frac{n}{2} \end{aligned} \tag{2.20}$$

In Table 2.1 the asymptotic cost of the different methods is shown. The asymptotic cost have been calculated taking the leader terms of the total number of operations. The detailed number of operations can be found in Annex A.

The results show that the complexity of MGE is very close to HR and is very low compared with the other methods. In particular, compared with MSGR, MGE reduce the number of operations in a 33 %. Besides, MGE, as well as MSGR, avoids the use of square roots.

A deeper discussion about complexity must take into account not only the number of operations, complexity can be reduced with parallel implementations. In that sense, a parallel VLSI architecture for matrix inversion using MGS was proposed in [3] and a systolic array implementation for GR and MSQR were proposed in [7] and [5] respectively. The implementation of GR and MSGR in systolic arrays allows the computation of more operations in parallel than the proposed structure for MGS. As the computation of MGE follows a similar idea than MSGR, it can also be implemented using systolic arrays. A fully parallel implementation of MGE using systolic arrays is presented in Section 2.6.

Regarding CORDIC, the variable p stands for the number of iterations in the CORDIC functions; it is known that generally one additional bit of accuracy is produced for each iteration [4]. Using this method the number of operations is higher than using the others, but with the advantage that

all the operations are shifts and additions, much more simple operations to implement in hardware than multiplications, divisions and square roots. The CORDIC algorithm is a good choice when no hardware multiplier is available, but if multipliers are available, the CORDIC algorithm is less efficient, as needs more iterations to convergence and occupy more area.

2.3 Numerical properties

In this section, the stability and accuracy of the computation of a matrix inverse using QR Decomposition is studied. If we denote the QR Decomposition affected by rounding errors as

$$\tilde{\mathbf{A}} = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}, \tag{2.21}$$

then the errors in the computation of $\mathbf{A^{-1}}$ can be written as

$$\tilde{\mathbf{A}}^{\mathbf{-1}} = \tilde{\mathbf{R}}^{\mathbf{-1}}\tilde{\mathbf{Q}}^{\mathbf{-1}} = \mathbf{A^{-1}} + \delta\mathbf{A^{-1}}. \tag{2.22}$$

In [8], it is proved that if a backward stable algorithm is used to solve a problem f with condition number κ, the relative errors satisfy

$$\frac{||\tilde{f}(x) - f(x)||}{||f(x)||} = O(\kappa(x)\epsilon_{machine}), \tag{2.23}$$

where the condition number is defined as

$$\kappa(\mathbf{A}) = ||\mathbf{A}||||\mathbf{A^{-1}}|| = \frac{\lambda_{max}}{\lambda_{min}}, \tag{2.24}$$

and λ_{max} and λ_{min} are the maximum and minimum eigenvalues respectively.

So, if we prove that the inverse of a matrix using QR Decomposition is backward stable, then the relative error of the inverse is upper-bound by

$$\frac{||\delta\mathbf{A^{-1}}||}{||\mathbf{A^{-1}}||} \leq C\kappa(\mathbf{A})\epsilon_{machine}. \tag{2.25}$$

For a QR Decomposition computed by Householder Reflectors, Givens Rotations, CORDIC Givens Rotations and Modified Gram-Schmidt, the factors $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ satisfy

$$\tilde{\mathbf{Q}}\tilde{\mathbf{R}} = \mathbf{A} + \delta\mathbf{A}, \frac{||\delta\mathbf{A}||}{||\mathbf{A}||} = O(\epsilon_{machine}) \tag{2.26}$$

due to the orthogonality of $\mathbf{Q}$. However, Gaussian Elimination is known to be numerical unstable for certain matrices. In [8] a deep discussion about the stability issues of Gaussian Elimination was done. The conclusion of the discussion was that the stability of a Gaussian Elimination decomposition with partial pivoting can not be assure for all matrices, but the set of matrices that can lead to stability problems in GE are so rarely found in real applications that Gaussian Elimination can be considered as backward stable when a partial pivoting is done. Here, we assume that Gaussian Elimination is backward stable. Regarding MSGR, a discussion about stability has not been done and can not be assure.

To prove that the inverse of a matrix using QR Decomposition is backward stable we must prove that

$$\frac{||\delta \mathbf{A^{-1}}||}{||\mathbf{A^{-1}}||} = O(\epsilon_{machine}) \tag{2.27}$$

We consider that the errors affecting the inverse are caused by the errors in the computation and inverse of $\mathbf{R}$ and $\mathbf{Q}$, so

$$\begin{aligned} \mathbf{A^{-1}} + \delta \mathbf{A^{-1}} = (\mathbf{R^{-1}} + \delta \mathbf{R^{-1}})(\mathbf{Q^{-1}} + \delta \mathbf{Q^{-1}}) = \\ = \mathbf{R^{-1}Q^{-1}} + \delta \mathbf{R^{-1}Q^{-1}} + \mathbf{R^{-1}}\delta \mathbf{Q^{-1}} + \delta \mathbf{R^{-1}}\delta \mathbf{Q^{-1}}. \end{aligned} \tag{2.28}$$

Considering the term $\delta \mathbf{R^{-1}}\delta \mathbf{Q^{-1}}$ negligible we obtain

$$\delta \mathbf{A^{-1}} = \delta \mathbf{R^{-1}Q^{-1}} + \mathbf{R^{-1}}\delta \mathbf{Q^{-1}}, \tag{2.29}$$

so the norm of $\delta \mathbf{A^{-1}}$ is

$$||\delta \mathbf{A^{-1}}|| = ||\delta \mathbf{R^{-1}Q^{-1}} + \mathbf{R^{-1}}\delta \mathbf{Q^{-1}}|| \leq ||\delta \mathbf{R^{-1}}||||\mathbf{Q^{-1}}|| + ||\mathbf{R^{-1}}||||\delta \mathbf{Q^{-1}}||, \tag{2.30}$$

and

$$\frac{||\delta \mathbf{A^{-1}}||}{||\mathbf{Q^{-1}}||||\mathbf{R^{-1}}||} \leq \frac{||\delta \mathbf{R^{-1}}||||\mathbf{Q^{-1}}||}{||\mathbf{Q^{-1}}||||\mathbf{R^{-1}}||} + \frac{||\mathbf{R^{-1}}||||\delta \mathbf{Q^{-1}}||}{||\mathbf{Q^{-1}}||||\mathbf{R^{-1}}||} = \frac{||\delta \mathbf{R^{-1}}||}{||\mathbf{R^{-1}}||} + \frac{||\delta \mathbf{Q^{-1}}||}{||\mathbf{Q^{-1}}||}. \tag{2.31}$$

The computation of $\mathbf{R^{-1}}$ is done using backward substitution, which was also proved to be backward stable in [8], so

$$\frac{||\delta \mathbf{R^{-1}}||}{||\mathbf{R^{-1}}||} = O(\kappa(\mathbf{A})\epsilon_{machine}). \tag{2.32}$$

Regarding $\mathbf{Q^{-1}}$, in HR, GR, CORDIC, MSGR and MGE the algorithm gives us directly $\mathbf{Q^{-1}}$ and the orthogonal errors are not affecting, only round-off errors are present. So, $\mathbf{Q^{-1}}$ is also backward stable

$$\frac{||\delta \mathbf{Q^{-1}}||}{||\mathbf{Q^{-1}}||} = O(\epsilon_{machine}). \tag{2.33}$$

In the case of Modified Gram-Schmidt, the algorithm gives us $\mathbf{Q}$ and we calculate $\mathbf{Q^{-1}}$ as $\mathbf{Q^H}$. In [3] it is shown that the loss of orthogonality is

$$\frac{||\delta \mathbf{Q^{-1}}||}{||\mathbf{Q^{-1}}||} = \frac{||\delta \mathbf{Q^H}||}{||\mathbf{Q^H}||} = O(\kappa(\mathbf{A})\epsilon_{machine}) \tag{2.34}$$

Now, taking (2.31), (2.32), (2.33) and (2.34) we can say that for HR, GR, CORDIC GR, MGE and MSGR

$$\frac{||\delta \mathbf{A^{-1}}||}{||\mathbf{Q^{-1}}||||\mathbf{R^{-1}}||} \leq \kappa(\mathbf{A})C_1\epsilon_{machine} + C_2\epsilon_{machine} = (\kappa(\mathbf{A})C_1 + C_2)\epsilon_{machine}, \tag{2.35}$$

and for MGS

$$\frac{||\delta \mathbf{A^{-1}}||}{||\mathbf{Q^{-1}}||||\mathbf{R^{-1}}||} \leq \kappa(\mathbf{A})C_1\epsilon_{machine} + C_3\kappa(\mathbf{A})\epsilon_{machine} = \kappa(\mathbf{A})(C_1 + C_3)\epsilon_{machine}. \tag{2.36}$$

Equations (2.35) and (2.36) does not assure the backward stability of $\mathbf{A^{-1}}$. Only in the case of orthogonal $\mathbf{Q}$ we can affirm that

$$\frac{||\delta \mathbf{A^{-1}}||}{||\mathbf{Q^{-1}}||||\mathbf{R^{-1}}||} = \frac{||\delta \mathbf{A^{-1}}||}{||\mathbf{A^{-1}}||}. \tag{2.37}$$

This is always true for HR, GR, CORDIC GR and MGS, but for MSGR and MGE we can not assure the backward stability of the inverse. However, as shown in [8], in real applications is very difficult to find a system when Gaussian Elimination is not stable, so we can say that even if the stability is not completely assure, in most applications the inverse is stable.

2.4 Floating Points Results

In this section, the relative errors of $\mathbf{A^{-1}}$, $\mathbf{R^{-1}}$ and $\mathbf{Q^H}$ for a set of matrices with increasing condition number are presented. The dot-dashed line represent the upper-bound value $\epsilon_{machine}$ and the dashed line the upper-bound

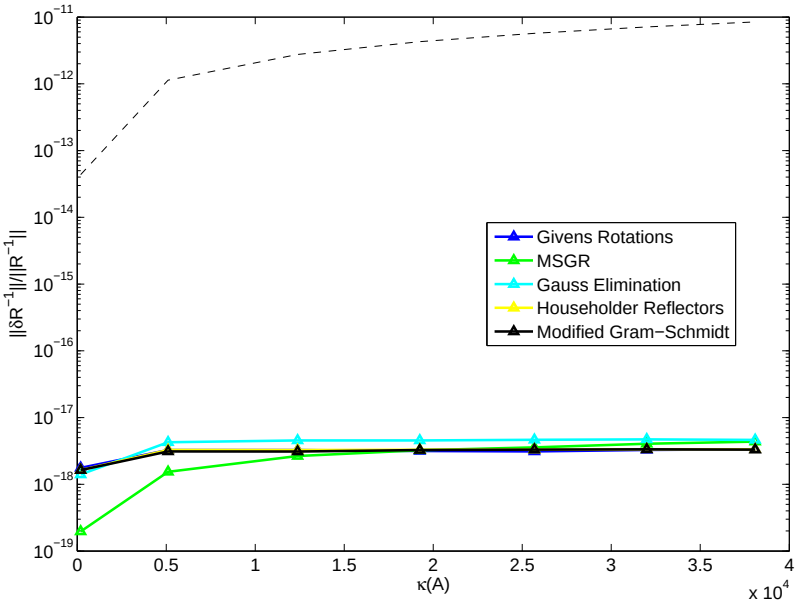


Figure 2.1: Relative errors of $\mathbf{R^{-1}}$ using double precision floating points

value $\kappa(\mathbf{A})\epsilon_{machine}$. The results have been obtained using MATLAB double precision, which have $\epsilon_{machine} = 2.2204 \cdot 10^{-16}$.

As shown in Figures 2.1 to 2.3, the relative errors of $\mathbf{A^{-1}}$, $\mathbf{R^{-1}}$ and $\mathbf{Q^H}$ are below the upper-bound values derived above and, in the case of $\mathbf{Q^H}$, also show that the orthogonalization errors of HR and GR are upper-bound by $\epsilon_{machine}$. The large orthogonalization errors in MGS are making the algorithm the less accurate whereas the most accurate algorithm is MGE.

2.5 Fixed Point Results

The analysis in Section 2.3 is valid for floating point arithmetic and can not be easily extrapolate to fixed points. Hence, here we repeat the simulations using fixed points to find the upper-bounds empirically. The simulations have been done using 64, 32 and 16 bits and normalizing the input to avoid over/under-flows.

2.5.1 64 bits fixed points

In Figure 2.4 we can observe that MGS have less errors than all the other methods. However, the orthogonalization errors (Fig.2.5) introduce an error

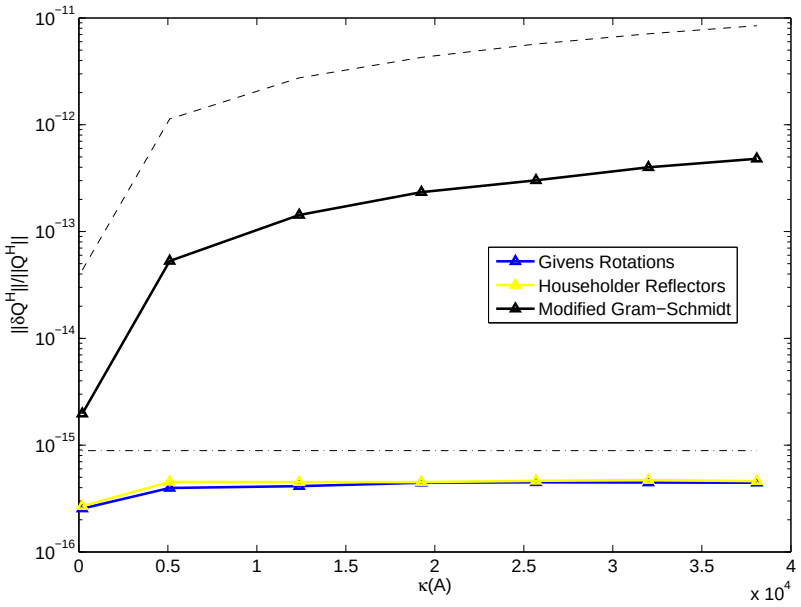


Figure 2.2: Relative errors of $\mathbf{Q^H}$ using double precision floating points

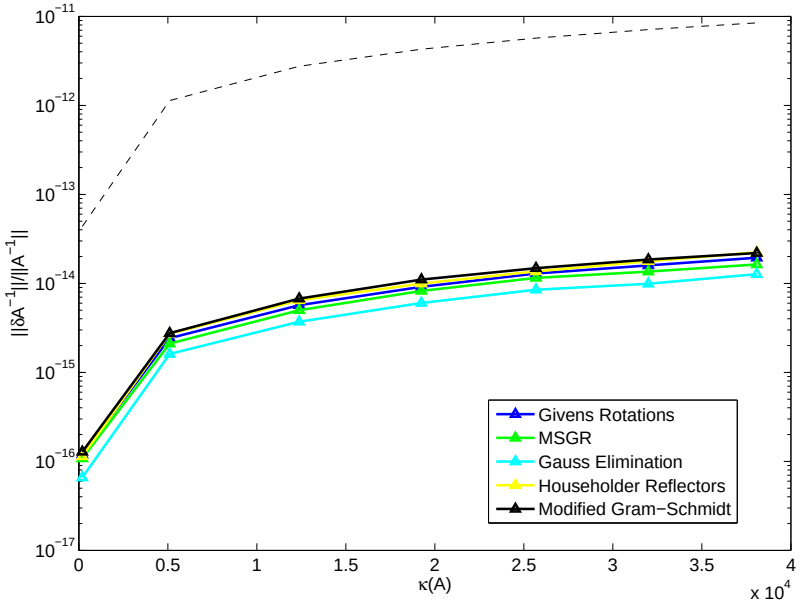


Figure 2.3: Relative errors of $\mathbf{A^{-1}}$ using double precision floating points

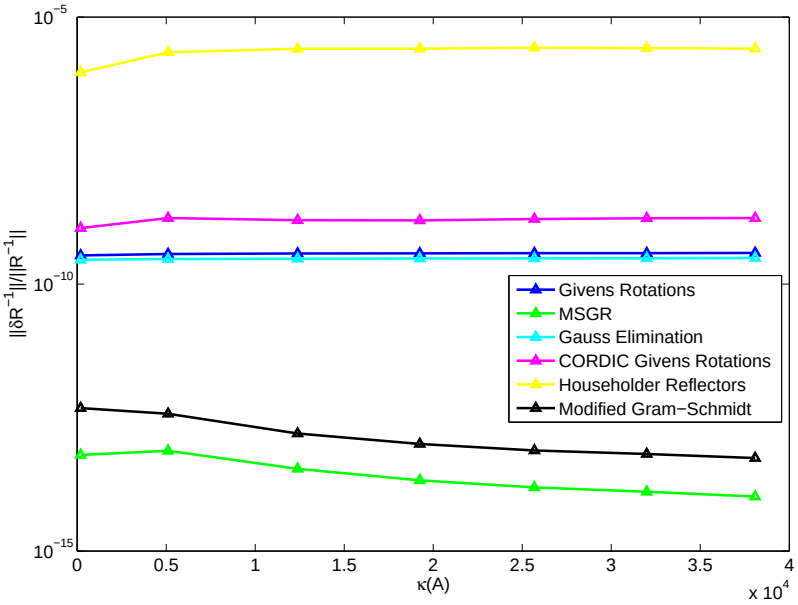


Figure 2.4: Relative errors of $\mathbf{R^{-1}}$ using 64-bits fixed points

in the computation of the matrix inverse that is not present in the other methods, as commented above. For this reason, MGS is the method that shows a larger error. In Fig. 2.6 the results show that MGE is the method with less error. All the methods show a similar dependence of the relative error of $\mathbf{A^{-1}}$ with the condition number.

2.5.2 32 bits fixed points

Using a 32-bits representation similar results were obtained. Figures 2.7 to 2.9 show that again MGS have less errors in the computation of $\mathbf{R^{-1}}$ but the orthogonalization errors are also limiting the performance. MSGR is showing less errors in $\mathbf{R^{-1}}$, but this is not reflected in $\mathbf{A^{-1}}$ because the computation of $\mathbf{Q^{-1}}$ is introducing a larger error. MGE is again the method with less error in $\mathbf{A^{-1}}$.

2.5.3 16 bits fixed points

The results obtained in Fig. 2.10 and Fig. 2.11 are the same as using 64 and 32 bits. In Fig. 2.12, however, all the methods show the same error except GR and HR, that show a larger error.

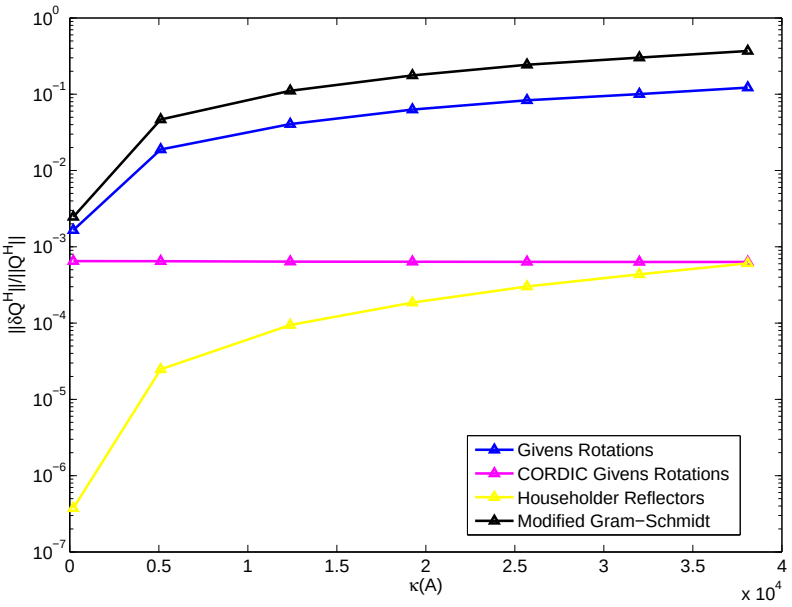


Figure 2.5: Relative errors of $\mathbf{Q^H}$ using 64-bits fixed points

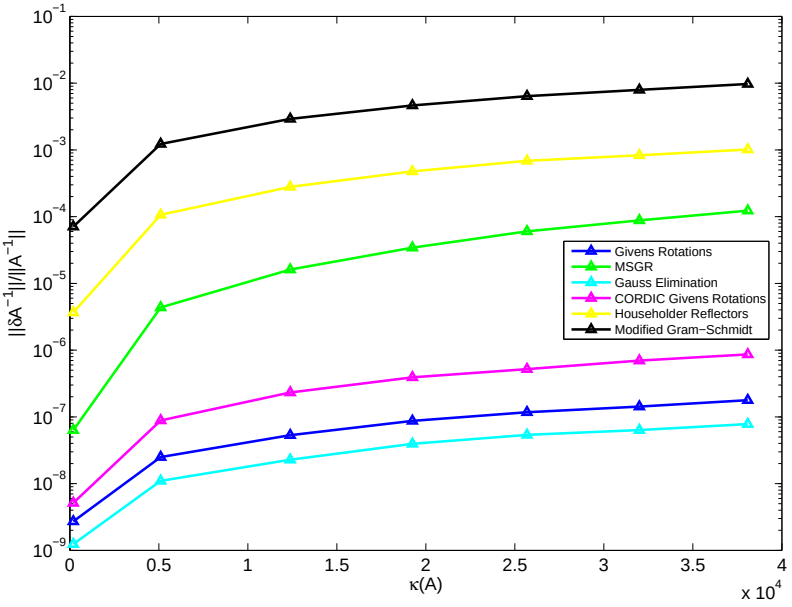


Figure 2.6: Relative errors of $\mathbf{A^{-1}}$ using 64-bits fixed points

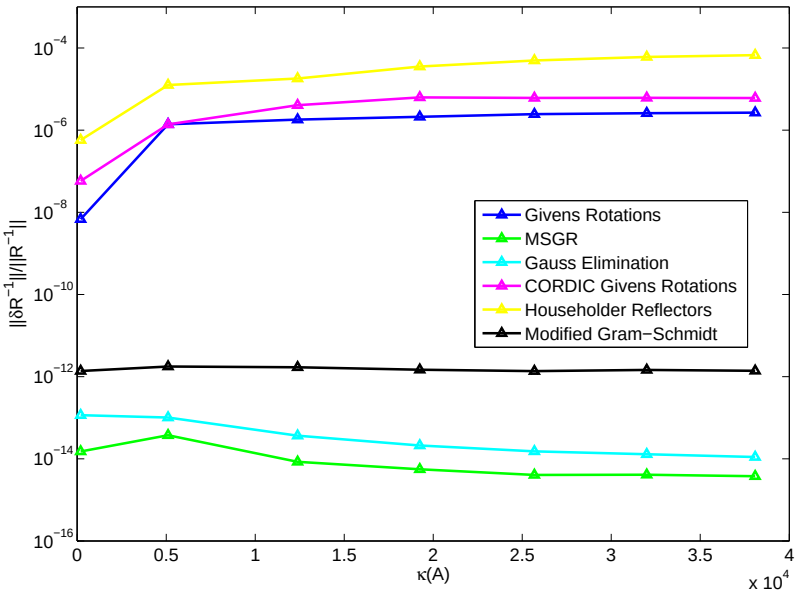


Figure 2.7: Relative errors of $\mathbf{R^{-1}}$ using 32-bits fixed points

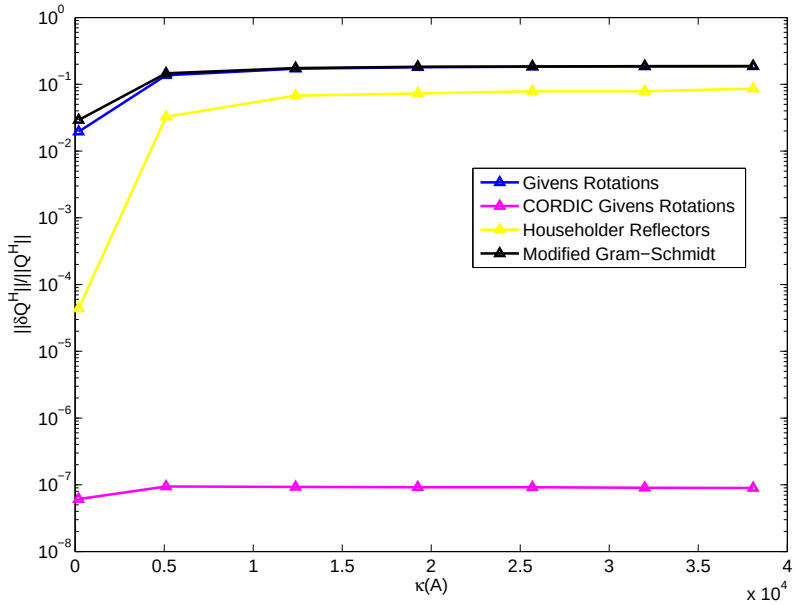


Figure 2.8: Relative errors of $\mathbf{Q^H}$ using 32-bits fixed points

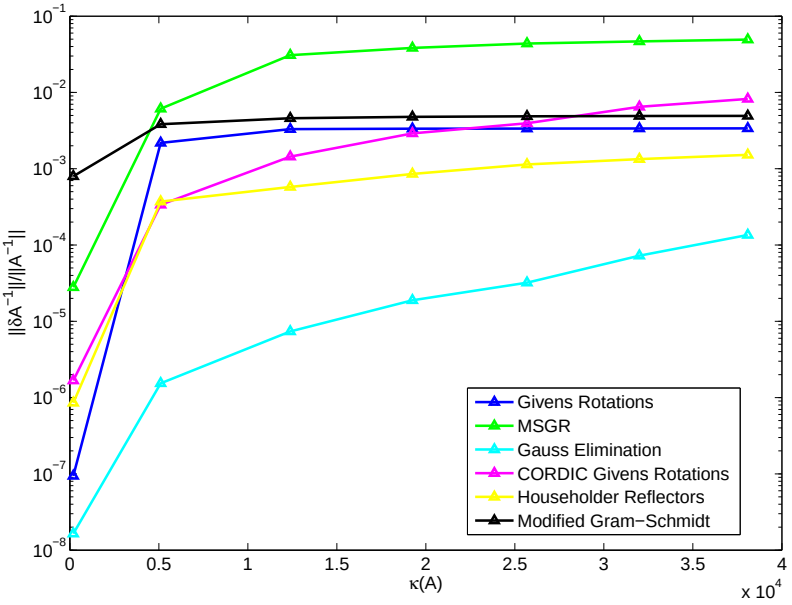


Figure 2.9: Relative errors of $\mathbf{A^{-1}}$ using 32-bits fixed points

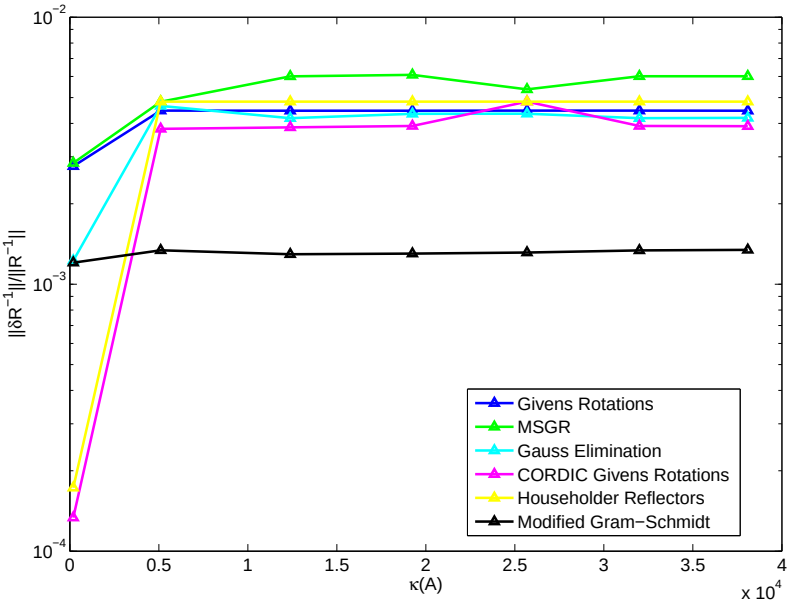


Figure 2.10: Relative errors of $\mathbf{R^{-1}}$ using 16-bits fixed points

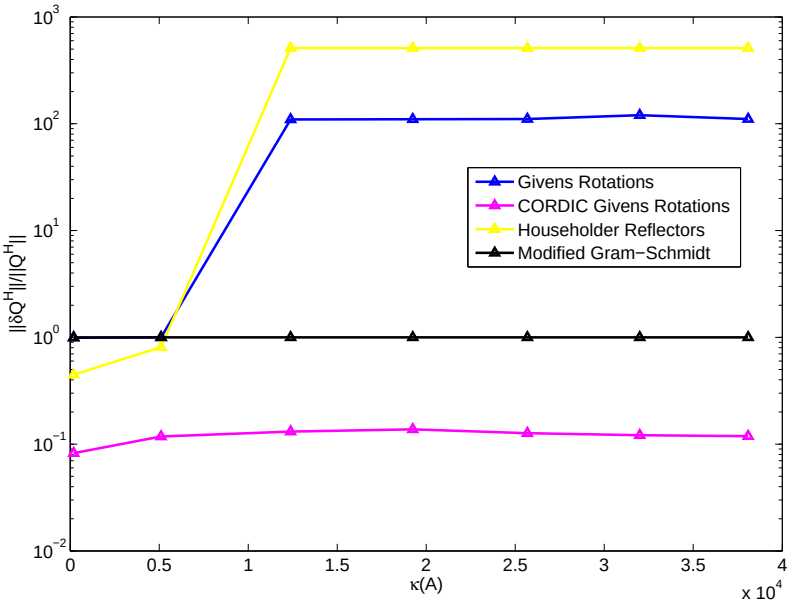


Figure 2.11: Relative errors of $\mathbf{Q^H}$ using 16-bits fixed points

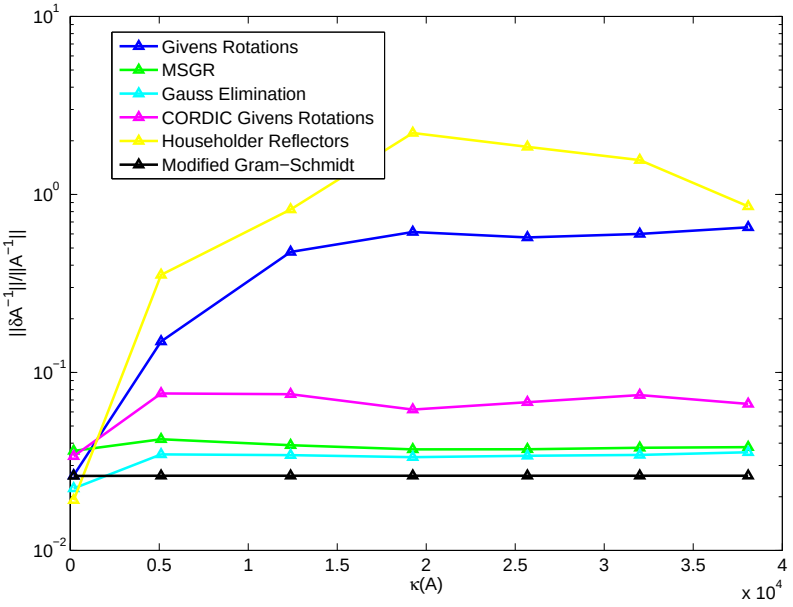
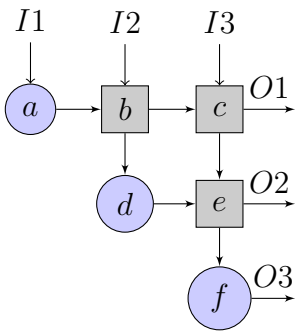


Figure 2.12: Relative errors of $\mathbf{A^{-1}}$ using 16-bits fixed points



(a)

x	x	x
x	x	x
x	x	1
x	0	0
0	1	0
0	0	r_{33}
1	r_{23}	r_{32}
r_{13}	r_{22}	r_{31}
r_{12}	r_{21}	0
r_{11}	0	0

(b)

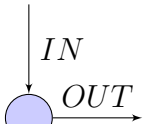
Figure 2.13: (a) Systolic array for the computation of QR using Gaussian Elimination (3×3 matrix), (b) Inputs of the systolic array.

2.6 Implementation of MGE

A completely full implementation of Givens Rotations using systolic arrays was proposed in [7], where the diagonal blocks compute the rotation matrices and the rest of blocks perform the rotation of the rows. In a similar way, MSGR can also be compute using systolic arrays [5]. In this case, the rotated elements are propagated and the rotation matrix is stored in the local memory of the blocks. To compute the rotation parameters, the diagonal elements of the input matrix must be tagged with a flag. When a diagonal element enter a block as an input, the rotation parameters are computed and store in the local memory of the block. When an element that is not tagged enter as an input the rotation is perform. Using that approach the data is flowing to the right and the final results are the outputs of the rightmost elements.

A similar structure is proposed here to compute the QR Decomposition using Gaussian Elimination (Fig. 2.13). The second approach was chosen as less blocks are needed and it has a considerably area saving. The blocks and the operations perform inside them are shown in Fig. 2.14 and Fig. 2.15.

The circle block just delay the data until the other needed inputs are available. In the square blocks, the rotation parameters are computed when there is a diagonal element as input and the rotation is perform when is off-diagonal. A partial pivoting to assure stability is also done. The data flow is the same as in the computation of MSGR, so an *Invert-and-Multiply Array* as presented in [5] can be use to compute the inverse of the input matrix after the decomposition.

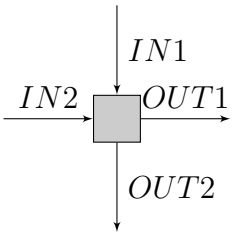


(a)

IN = OUT

(b)

Figure 2.14: Detail of the Circle Block. (a) Inputs and Outputs of the block. (b) Code of the block.



if $flag$ ***equal to*** 1 **then**
  if $IN2$ ***equal to*** 0 **then**
    $A = 1$
    $B = 0$
    $C = 0$
    $D = 1$
  else
    $A = 0$
    $B = 1$
    $C = 1$
    $D = -IN1/IN2$
  end
end
$OUT1 = A \cdot IN1 + B \cdot IN2$
$OUT2 = C \cdot IN1 + D \cdot IN2$

Figure 2.15: Detail of the Square Block. (a) Inputs and Outputs of the block. (b) Code of the block.

2.7 Conclusions

In this first part of the thesis we have analyse and compared different QR Decompositions methods applied to matrix inversion and a new method based in Gaussian Elimination has been proposed, called Modified Gaussian Elimination. The results have shown that the proposed method has less complexity than the others and avoid the use of complex operations such as square roots. In terms of numerical properties, the stability of MGE can be assure in real applications and have a better accuracy than the other methods due to the simple operations required. Besides, a fully parallel implementation of MGE applied to matrix inversion have been proposed to reduce the latency of the system. On the other hand, the matrix $\mathbf{Q}$ obtained using MGE is not orthogonal, so in applications when the orthogonality of $\mathbf{Q}$ is required the method can not be applied. In that case, the method that show better results in terms of accuracy is CORDIC Givens Rotations, but at the expense of more latency and area cost. When low latency or small area is required, the best option has been proved to be Givens Rotations.

Chapter 3

Kalman Filter

Kalman filters have been widely applied in many applications since was developed in 1960 by Kalman [9]. The Kalman Filter minimize the MSE of a signal embedded in noise using a linear recursion, where the signal is characterized by a model. The Kalman filter is an optimal MMSE estimator when the signals and noise are jointly Gaussian and the model is linear. However, when the model or the relation between states and measurements is non-linear, the filter is not optimal and the transition and observation functions have to be approximated. The Extended Kalman Filter is the most used version of the filter in the non-linear case. The EKF do a first order approximation of the non-linear functions and apply the linearized functions to the Kalman filter equations. The EKF have been deeply studied in the literature and have been shown to be very unstable due to the effect of linearization errors [10]. To improve the approximation of the non-linear functions, several alternatives have been proposed, such as the use of second order approximations and the unscented filters. The second order approximations have proved to increase considerably the complexity without improving the performance [10]. On the other hand, the unscented filters do a third order approximation of the non-linear functions, increase considerably the accuracy and stability of the filter and have a complexity of the same order than the EKF [11]. Square root versions of all the previous filters have been also developed to assure the symmetry and semi-positiveness of the covariance matrices. In this second part of the thesis, an analysis of different non-linear Kalman Filters is done in terms of accuracy, stabilty and complexity.

3.1 Vehicle Model

The goal of this section is to introduce the model that will be used in the following sections. The simulations are done considering the model given in [12]. In that book, the model considered is the tracking of a vehicle with constant velocity perturbed only by external forces, i.e. wind gusts. These perturbations are modelled as noise inputs

$$\begin{aligned} v_{x,i} &= v_{x,i-1} + u_{x,i} \\ v_{y,i} &= v_{y,i-1} + u_{y,i} \end{aligned} \tag{3.1}$$

Then, we can model the position of the vehicle as

$$\begin{aligned} r_{x,i} &= r_{x,i-1} + v_{x,i-1}\Delta \\ r_{y,i} &= r_{y,i-1} + v_{y,i-1}\Delta \end{aligned} \tag{3.2}$$

where Δ is the time interval between samples and $\mathbf{u}$ is the process noise. The states of the filter are the two components of the position and velocity

$$\mathbf{x_i} = \begin{pmatrix} r_{x,i} \\ r_{y,i} \\ v_{x,i} \\ v_{y,i} \end{pmatrix}, \tag{3.3}$$

so the states follow the linear recursion

$$\mathbf{x_{i+1}} = \mathbf{F}\mathbf{x_i} + \mathbf{u_i}, \tag{3.4}$$

where $\mathbf{F}$ is the state transition function

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.5}$$

The process noise is assume to be Gaussian, time invariant with zero mean and covariance matrix

$$\mathbf{Q} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_u^2 & 0 \\ 0 & 0 & 0 & \sigma_u^2 \end{pmatrix}. \tag{3.6}$$

In most tracking applications, for instance in radar, range and bearing of the position are measured instead of the coordinates of the position

$$\mathbf{y_i} = \begin{pmatrix} R_i \\ \beta_i \end{pmatrix} \tag{3.7}$$

where

$$\begin{aligned} R_i &= \sqrt{r_{x,i}^2 + r_{y,i}^2} + v_{R,i}, \\ \beta_i &= arctan(\tfrac{r_{y,i}}{r_{x,i}}) + v_{\beta,i}, \end{aligned} \tag{3.8}$$

and $\mathbf{v}$ stands for the measurement noise, which is assume to be Gaussian, time invariant, uncorrelated with the process noise, zero mean and with covariance

$$\mathbf{R} = \begin{pmatrix} \sigma_R^2 & 0 \\ 0 & \sigma_\beta^2 \end{pmatrix}. \tag{3.9}$$

The initial state of the filter is also assume to be Gaussian and uncorrelated with the process and measurement noises.

It is clear that the relation between the measurements and the states is highly non-linear

$$\mathbf{y_i} = \mathbf{h}(\mathbf{x_i}) + \mathbf{v_i}, \tag{3.10}$$

where $\mathbf{h}$ is the observation function, so non-linear versions of the Kalman filter have to be applied.

3.2 Extended Kalman Filters

The EKF makes a linearization of the observation function about the estimate of $\mathbf{x_i}$ based on the previous data $\mathbf{x_{i|i\text{-}1}}$ taking the first order approximation of the Taylor expansion, which yields in the following approximation of the observation matrix

$$\mathbf{H_i} = \frac{\partial \mathbf{h}}{\partial \mathbf{x_i}}\Big|_{\mathbf{x}_i = \mathbf{x_{i|i\text{-}1}}}. \tag{3.11}$$

For the model considered here, the approximation results in an observation matrix of the form

$$\mathbf{H_i} = \begin{pmatrix} \frac{r_{x,i}}{\sqrt{r_{x,i}^2+r_{y,i}^2}} & \frac{r_{y,i}}{\sqrt{r_{x,i}^2+r_{y,i}^2}} & 0 & 0 \\ \frac{-r_{y,i}}{r_{x,i}^2+r_{y,i}^2} & \frac{r_{x,i}}{r_{x,i}^2+r_{y,i}^2} & 0 & 0 \end{pmatrix}. \tag{3.12}$$

The EKF applies the linearization of the observation matrix to the well-known kalman filter equations [12]

$$\mathbf{x_{i|i-1}} = \mathbf{F_i x_{i-1|i-1}}, \tag{3.13}$$

$$\mathbf{P_{i|i-1}} = \mathbf{F_i P_{i-1|i-1} F_i^H} + \mathbf{Q}, \tag{3.14}$$

$$\mathbf{K_i} = \mathbf{P_{i|i-1} H_i^H R_{e,i}^{-1}}, \tag{3.15}$$

$$\mathbf{R_{e,i}} = \mathbf{H_i P_{i|i-1} H_i^T} + \mathbf{R}, \tag{3.16}$$

$$\mathbf{x_{i|i}} = \mathbf{x_{i|i-1}} + \mathbf{K_i}(\mathbf{y_i} - \mathbf{h}(\mathbf{x_{i|i-1}})), \tag{3.17}$$

$$\mathbf{P_{i|i}} = (\mathbf{I} - \mathbf{K_i H_i})\mathbf{P_{i|i-1}}. \tag{3.18}$$

The implementation of the EKF is more complex than the Linear Kalman Filter. For linear time invariant models, the computation of the Kalman Gain and the covariance of the estimates are not dependent on the measurements or estimates and can be precomputed to simplify the computation. However, for non-linear models, matrices $\mathbf{F}$ and $\mathbf{H}$ are the Jacobian of the state transition and measurements functions. These matrices have to be computed from the estimates in every iteration and can not be precomputed. That increases the complexity because the filter must compute in every iteration more and more complex operations, including a matrix inversion in the Kalman Gain equation (3.15). To reduce the computational load, several implementations have been proposed. In the next sections, the implementation of the filter using QR to compute the matrix inverse and an LDL based implementation are analysed.

3.2.1 QR-EKF

The QR-EKF directly apply the EKF using the QR Decomposition to compute the matrix inversion in the Kalman Gain equation (3.15). A complete description of the algorithm is given in Alg. 8. The method used to compute the matrix inversion is MGE, since is the method with less complexity and have good numerical properties. The complexity of the filter is given in Table 3.1 in terms of number of operations. The results show the number of additions, subtractions, multiplications, divisions and square roots. Additions

Algorithm 8: QR-EKF

Time update equations

1 $\mathbf{x_{i|i\text{-}1}} = \mathbf{F}\mathbf{x_{i\text{-}1|i\text{-}1}}$

2 $\mathbf{P_{i|i\text{-}1}} = \mathbf{F}\mathbf{P_{i\text{-}1|i\text{-}1}}\mathbf{F^H} + \mathbf{Q}$

Linearization

3 $r = \mathbf{x^2_{i|i\text{-}1}[1]} + \mathbf{x^2_{i|i\text{-}1}[2]}$

4 $b = atan(\frac{\mathbf{x_{i|i\text{-}1}[2]}}{\mathbf{x_{i|i\text{-}1}[1]}})$

5 $\mathbf{h_i} = \begin{pmatrix} \sqrt{r} & b \end{pmatrix}$

6 $\mathbf{H_i} = \begin{pmatrix} \frac{\mathbf{x_{i|i\text{-}1}[1]}}{\sqrt{r}} & \frac{\mathbf{x_{i|i\text{-}1}[2]}}{\sqrt{r}} & 0 & 0 \\ \frac{-\mathbf{x_{i|i\text{-}1}[2]}}{r} & \frac{\mathbf{x_{i|i\text{-}1}[1]}}{r} & 0 & 0 \end{pmatrix}$

Measurement update equations

7 $\mathbf{R_{e,i}} = \mathbf{H_i}\mathbf{P_{i|i\text{-}1}}\mathbf{H_i^H} + \mathbf{R}$

QR Decomposition of $\mathbf{R_{e,i}}$

8 $\mathbf{R_{e,i}} = \mathbf{Q_{R_{e,i}}}\mathbf{R_{R_{e,i}}}$

Inverse of $\mathbf{R_{e,i}}$ using QR Decomposition

9 $\mathbf{R^{\text{-}1}_{e,i}} = \mathbf{R^{\text{-}1}_{R_{e,i}}}\mathbf{Q^{\text{-}1}_{R_{e,i}}}$

10 $\mathbf{K_i} = \mathbf{P_{i|i\text{-}1}}\mathbf{H_i^H}\mathbf{R^{\text{-}1}_{e,i}}$

11 $\mathbf{x_{i|i}} = \mathbf{x_{i|i\text{-}1}} + \mathbf{K_i}(\mathbf{y_i} - \mathbf{h})$

12 $\mathbf{P_{i|i}} = (\mathbf{I} - \mathbf{K_i}\mathbf{H_i})\mathbf{P_{i|i\text{-}1}}$

	QR-EKF
Additions	$\mathbf{3n^3 + 4n^2p + np^2} - 2n^2 + p^2 - 3np - n$
Substractions	$\frac{\mathbf{5p^3}}{\mathbf{6}} + n^2 - \frac{p^2}{2} + \frac{2p}{3}$
Multiplications	$\mathbf{3n^3} + \frac{\mathbf{5p^3}}{\mathbf{6}} + \mathbf{3n^2p + 2np^2} + n^2 - \frac{p^2}{2} + np - \frac{p}{3}$
Divisions	$\frac{p^2}{2} - \frac{p}{2}$
Square Roots	0
Others	Linearization

Table 3.1: Number of operations of QR-EKF

and subtractions are very simple to compute in processors while square root are very complex and should be avoid when possible.

The number of operations show that the filter is square root free and all the divisions come from the matrix inversion. The linearization of the measurement function requires the computations of the Jacobian of the function, which could add a lot of complexity to the filter. In the case of the example considered, the computation of the Jacobian and the application of the non-linear function to the estimates, (3.17), requires 2 multiplications, 1 addition, 4 divisions, 1 square root and 1 atan (lines 3 to 6 of Alg. 8).

3.2.2 LDL-EKF

Here, an square root version of the EKF is introduced. Square root filters have proved to have good numerical properties since they assure the symmetry of the error covariance matrices in presence of rounding errors [13]. Square root filters propagate the square factors of the matrices instead of the whole matrix in order to assure its symmetry. Letting $\mathbf{P}$ denote a positive-definite matrix, a square-root factor of $\mathbf{P}$ is any square matrix $\mathbf{L}$ such that $\mathbf{P} = \mathbf{L^H L}$. In the case of the filter considered here, the covariance matrix $\mathbf{P}$ is decomposed as

$$\mathbf{P} = \mathbf{L D L^H} = (\mathbf{L D^{\frac{1}{2}}})(\mathbf{L D^{\frac{1}{2}}})^{\mathbf{H}}, \tag{3.19}$$

and the factors $\mathbf{L}$ and $\mathbf{D^{\frac{1}{2}}}$ are propagated. The algorithm to compute the LDL Decomposition is given in Alg. 9.

The drawback of these filters is complexity since computation of square roots is needed. An elegant solution to this problem was proposed in [14]. The paper proposed a new algorithm to the measurement update equations of the square root filter that avoids the computation of square roots. The algorithm is given in Alg. 10. A complete derivation of the filter can be found in [14].

Algorithm 9: LDL Decomposition

1 $\mathbf{L} = \mathbf{I}_{n\times n}$
2 $\mathbf{A2} = diag(\mathbf{A})$
3 $\mathbf{D} = diag(\mathbf{A})$
4 **for** $j = 1$ ***to*** n **do**
5 **for** $i = j + 1$ ***to*** n **do**
6 $d = 0$
7 $l = 0$
8 **for** $k = 1$ ***to*** $j - 1$ **do**
9 $d = d + \mathbf{L}_{j,k}\mathbf{L}_{j,k}\mathbf{D}_k$
10 $l = l + \mathbf{L}_{i,k}\mathbf{D}_k\mathbf{L}_{j,k}$
11 **end**
12 $\mathbf{D}_j = \mathbf{A2}_j - d$
13 $\mathbf{L}_{i,j} = \mathbf{A}_{i,j} - l$
14 $\mathbf{L}_{i,j} = \mathbf{L}_{i,j}/\mathbf{D}_j$
15 **end**
16 $\mathbf{D}_n = \mathbf{A2}_n - \mathbf{L}_{n,1:n-1}\mathbf{L}'_{n,1:n-1}\mathbf{D}_{1:n-1}$
17 $\mathbf{D} = diag(\mathbf{D})$
18 **end**

It is clear that only the $\mathbf{L_{p,i}}$ an $\mathbf{D_{p,i}}$ terms of the covariance matrix need to be propagated to the next iteration. Using this algorithm, we are taking advantage of the good numerical properties of square root filters at the same time that we avoid the use of square roots in the computation of the measurement update equations. On the other hand, in the time update of the covariance matrix, it is necessary to take the square root of the diagonal matrix $\mathbf{D_{p,i}}$ because Modified Givens Rotations gives us directly the diagonal matrix, no its square root. However, the number of square roots required is very small, only n in every iteration of the filter. The number of operations for the LDL-EKF are given in Table 3.2.

We can observe that the SR-EKF increases considerably the complexity and introduce more divisions and square roots. The increase in complexity is mostly caused by the computation of the LDL's decomposition and the use of Modified Givens Rotations. However, a fully parallel structure was proposed in [14] to compute the measurement update. Using this structure, it is possible to compute the operations in lines 11 to 13 in parallel and reduce the computation time considerably.

Algorithm 10: LDL-EKF

Initialization

$\mathbf{R} = \mathbf{L_R D_R L_R^H}$

$\mathbf{P_{-1}} = \mathbf{L_p D_p L_p^H}$

Time update equations

$\mathbf{x_{i|i-1}} = \mathbf{F x_{i-1|i-1}}$

$\mathbf{M} = \begin{pmatrix} \mathbf{F L_{p,i-1} D_{p,i-1}^{\frac{1}{2}}} & \mathbf{Q^{\frac{1}{2}}} \end{pmatrix}$

$\mathbf{P_{i|i-1}} = \mathbf{M M^H}$

Linearization

$r = \mathbf{x_{i|i-1}^2[1]} + \mathbf{x_{i|i-1}^2[2]}$

$b = atan(\frac{\mathbf{x_{i|i-1}[2]}}{\mathbf{x_{i|i-1}[1]}})$

$\mathbf{h_i} = \begin{pmatrix} \sqrt{r} & b \end{pmatrix}$

$\mathbf{H_i} = \begin{pmatrix} \frac{\mathbf{x_{i|i-1}[1]}}{\sqrt{r}} & \frac{\mathbf{x_{i|i-1}[2]}}{\sqrt{r}} & 0 & 0 \\ \frac{-\mathbf{x_{i|i-1}[2]}}{r} & \frac{\mathbf{x_{i|i-1}[1]}}{r} & 0 & 0 \end{pmatrix}$

Measurement update equations

$\mathbf{P_{i|i-1}} = \mathbf{L_i D_i L_i^H}$

Form matrices $\mathbf{L_1}$ and $\mathbf{D_1}$

$\mathbf{L_1} = \begin{pmatrix} \mathbf{L_{R,i}} & \mathbf{H_i L_i} \\ 0 & \mathbf{L_i} \end{pmatrix}$; $\mathbf{D_1} = \begin{pmatrix} \mathbf{D_R} & 0 \\ 0 & \mathbf{D_i} \end{pmatrix}$

Rotate using Modified Givens Rotations (see Alg. 11)

$\begin{pmatrix} \mathbf{L_R} & \mathbf{H_i L_i} \\ 0 & \mathbf{L_i} \end{pmatrix} \begin{pmatrix} \mathbf{D_R} & 0 \\ 0 & \mathbf{D_i} \end{pmatrix}^{\frac{1}{2}} \ominus = \begin{pmatrix} \mathbf{L_{Re}} & 0 \\ \mathbf{K_i L_{Re}} & \mathbf{L_{p,i}} \end{pmatrix} \begin{pmatrix} \mathbf{D_{Re}} & 0 \\ 0 & \mathbf{D_{p,i}} \end{pmatrix}^{\frac{1}{2}}$

Extract $\mathbf{K_i}$, $\mathbf{L_{p,i}}$ and $\mathbf{D_{p,i}}$

$\mathbf{x_{i|i} = x_{i|i-1} + K_i(y_i - h_i)}$

Algorithm 11: Modified Givens Rotations

$\mathbf{R = A}$
$\mathbf{Q = I}_{n \times n}$
for *i = 1* ***to*** *m-1* **do**
  for *j = n* ***to*** *i + 1* **do**
    $p_1 = \mathbf{R}_{i,i}$
    $p_2 = \mathbf{R}_{i,j}$
    $d_{p,1} = \mathbf{D}_{i,i}$
    $d_{p,2} = \mathbf{D}_{j,j}$
    $d_{q,1} = p_1 p_1 d_{p,1} + p_2 p_2 d_{p,2}$
    $d_{q,2} = d_{p,1} d_{p,2} / d_{q,1}$
    $d = \begin{pmatrix} p_1 d_{p,1}/d_{q,1} & -p_2 & p_2 d_{p,2}/d_{q,1} & p_1 \end{pmatrix}$
    $\mathbf{D}_{i,i} = d_{q,1}$
    $\mathbf{D}_{j,j} = d_{q,2}$
    $\mathbf{R}_{:,[i,j]} = \mathbf{R}_{:,[i,j]}\mathbf{d}$
  end
end

	LDL-EKF
Additions	$\mathbf{6n^3 + p^3 + 3n^2p + 4np^2} - \frac{7n^2}{2} - \frac{p^2}{2} - np - \frac{3n}{2} - \frac{p}{2}$
Substractions	$\frac{\mathbf{n^3}}{\mathbf{3}} + \frac{\mathbf{p^3}}{\mathbf{6}} - \frac{n}{3} - \frac{2p}{3}$
Multiplications	$\frac{\mathbf{22n^3}}{3} + \frac{\mathbf{13p^3}}{\mathbf{6}} + \mathbf{6n^2p + 7np^2} + \frac{5n^2}{2} + \frac{3p^2}{2} + 4np - \frac{23n}{6} - \frac{11p}{3}$
Divisions	$\frac{5n^2}{2} + 2p^2 + 3np - \frac{3n}{2} - p$
Square Roots	n
Others	Linearization

Table 3.2: Number of operations of LDL-EKF

3.3 Unscented Kalman Filters

The flaws of the EKF come from the poor linearization perform by the filter. Several improvements in the approximation of the non-linear functions have been proposed, being the unscented transform the most interesting as improve the approximation up to the third order without increasing the complexity compared to the EKF. The unscented transform choose selected samples of the pdf, called sigma points, that completely characterized the mean and covariance of the pdf. Then, the sigma points are propagated through the non-linear function and the mean and covariance of the new sigma points is calculated. An illustration of the transformation is presented in Figure 3.1.

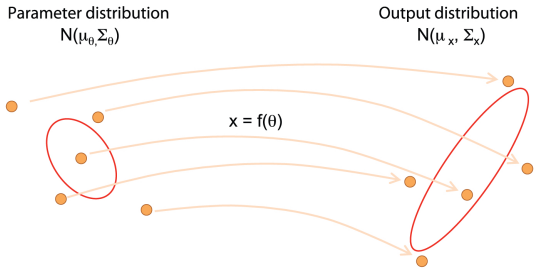


Figure 3.1: Unscented Transform. The Figure was obtained from [1]

That transform yields to a third order approximation of the two first moments of the new pdf in the case of a Gaussian distribution and a second order approximation for other distributions. Another advantage of the UKF is that avoids the computation of Jacobians and Hessians that in some applications can be very difficult or impossible to compute.

3.3.1 UKF

The UKF algorithm is given in Alg. 12. A complete derivation of the filter can be found in [15].

The number of operations for the UKF is given in Table 3.3. In the Table we can observe that the UKF requires more operations than the QR-EKF: it requires more additions, subtractions, multiplications, divisions and square roots. Besides, we are propagating all the sigma points through the non-linear function, so $2n + 1$ atan instead of 1 are needed using the unscented transform. That makes the filter more complex in terms of number and type of operations. Compared with the LDL-EKF, the filter requires less additions, subtractions, multiplications and divisions, but computes more square roots and atan.

Algorithm 12: UKF

Initialization of coeficients

κ such that $n + \kappa = 3$
$beta = 2$ for Gaussian distributions
$10^{-4} \leq \alpha \leq 1$
$\lambda = \alpha^2(n + k) - n$
$W_0^m = \lambda/(n + \lambda)$
$W_i^m = W_i^c = 1/(2(n + \lambda))$
$W_0^c = \lambda/(n + \lambda) + (1 - \alpha^2 + \beta)$

Calculate sigma points

Calculate Cholesky Decomposition of $(n + \lambda)P_{i-1|i-1}$ (see Alg. 13)

$\mathbf{S} = \sqrt{(n + \kappa)P_{i-1|i-1}}$

$\boldsymbol{\chi} = \begin{pmatrix} \mathbf{x}_{i|i-1} & \mathbf{x}_{i|i-1} + \mathbf{S}_{:,k} & \mathbf{x}_{i|i-1} - \mathbf{S}_{:,k} \end{pmatrix}$
$for\ k\ \epsilon\ \{1, ..., n\}$

Time update equations

Propagate through state transition function

$\boldsymbol{\chi}' = \mathbf{f}(\boldsymbol{\chi})$

Calculate Mean and Covariance of the states

$\mathbf{x_{i|i-1}} = \sum_{k=0}^{2n+1} W_k^m \boldsymbol{\chi}'_{\mathbf{k}}$

$\mathbf{P_{i|i-1}} = \sum_{k=0}^{2n+1} W_k^c (\boldsymbol{\chi}'_{\mathbf{k}} - \mathbf{x_{i|i-1}})(\boldsymbol{\chi}'_{\mathbf{k}} - \mathbf{x_{i|i-1}})^{\mathbf{H}}$

Propagate through observation function

$\boldsymbol{\zeta}' = \mathbf{h}(\boldsymbol{\chi})$

Calculate Mean of the predicted measurements

$\mathbf{y}'_{\mathbf{i}} = \sum_{k=0}^{2n+1} W_k^m \boldsymbol{\zeta}'_{\mathbf{k}}$

Measurement update equations

15 $\mathbf{P_{zz,i}} = \sum_{k=0}^{2n+1} W_k^c(\boldsymbol{\zeta}'_\mathbf{k} - \mathbf{y}'_\mathbf{i})(\boldsymbol{\zeta}'_\mathbf{k} - \mathbf{y}'_\mathbf{i})^\mathbf{H}$

16 $\mathbf{P_{xz,i}} = \sum_{k=0}^{2n+1} W_k^c(\boldsymbol{\chi}'_\mathbf{k} - \mathbf{x_{i|i\text{-}1}})(\boldsymbol{\zeta}'_\mathbf{k} - \mathbf{y}'_\mathbf{i})^\mathbf{H}$

QR Decomposition of $\mathbf{P_{zz,i}}$

17 $\mathbf{P_{zz,i}} = \mathbf{Q}_{\mathbf{P_{zz,i}}}\mathbf{R}_{\mathbf{P_{zz,i}}}$

Inverse of $\mathbf{P_{zz,i}}$ using QR Decomposition

18 $\mathbf{P^{\text{-}1}_{zz,i}} = \mathbf{R^{\text{-}1}_{P_{zz,i}}}\mathbf{Q^{\text{-}1}_{P_{zz,i}}}$

19 $\mathbf{K_i} = \mathbf{P_{xz,i}}\mathbf{P^{\text{-}1}_{zz,i}}$

20 $\mathbf{x_{i|i}} = \mathbf{x_{i|i\text{-}1}} + \mathbf{K_i}(\mathbf{y_i} - \mathbf{y_i}')$

21 $\mathbf{P_{i|i}} = \mathbf{P_{i|i\text{-}1}} - \mathbf{K_i}\mathbf{P_{zz,i}}\mathbf{K_i^H}$

Algorithm 13: Cholesky Decomposition

1 $\mathbf{R} = \mathbf{A}$
2 **for** $k = 1$ ***to*** n **do**
3 | **for** $j = k + 1$ ***to*** n **do**
4 | | $\mathbf{R}_{j,j:n} = \mathbf{R}_{j,j:n} - \mathbf{R}_{k,j:n}\mathbf{R}_{k,j}/\mathbf{R}_{k,k}$
5 | **end**
6 | $\mathbf{R}_{k,k:n} = \mathbf{R}_{k,k:n}/\sqrt{\mathbf{R}_{k,k}}$
7 **end**
8 **for** $i = 1$ ***to*** n **do**
9 | **for** $j = i$ ***to*** n **do**
10 | | $\mathbf{L}_{i,j} = \mathbf{R}_{i,j}$
11 | **end**
12 **end**
13 $\mathbf{L} = \mathbf{L}'$

	UKF
Additions	$\mathbf{4n^3 + 3n^2p + 4np^2} + 2p^2 + 2np + 6n + p + 1$
Substractions	$\frac{\mathbf{n^3}}{\mathbf{6}} + \frac{\mathbf{5p^3}}{\mathbf{6}} + 3n^2 - \frac{p^2}{2} + 6np + \frac{41n}{6} + \frac{11p}{3}$
Multiplications	$\frac{\mathbf{25n^3}}{\mathbf{6}} + \frac{\mathbf{5p^3}}{\mathbf{6}} + \mathbf{3n^2p + 4np^2} + 5n^2 + \frac{5p^2}{2} + 4np + \frac{35n}{6} + \frac{5p}{3} + 2$
Divisions	$n^2 + \frac{p^2}{2} - \frac{p}{2}$
Square Roots	$3n + 1$
Others	atan: $2n + 1$

Table 3.3: Number of operations of UKF

3.3.2 SR-UKF

In the UKF, a Cholesky Decomposition have to be computed in every iteration to obtain the sigma points. This decomposition is very costly and is increasing a lot the complexity of the filter, so a new version of the filter, called Square Root Unscented Kalman Filter, was proposed in [11] to avoid the computation of the Cholesky Decomposition every iteration. In this new approach, the Cholesky Factors of the covariance matrix are propagated and update instead of computing them in every iteration. The update of the Cholesky Factors is done using QR and a rank-one Cholesky update, as demonstrated in the cited paper. The propagation of the Cholesky Factors also assure the symmetry and semi-positiveness of the covariance matrices, reducing the symmetry errors that can produce the rounding errors. The SR-UKF algorithm is given in Alg. 14. A complete derivation of the filter can be found in [11].

The number of operations for the SR-UKF is given in Table 3.4. The algorithm does not require the computation of the matrix $\mathbf{Q}$ of the QR Decomposition, so the complexity of the decomposition can be reduced almost at half. However, the algorithm requires the orthogonality of $\mathbf{Q}$, so the MGE method can not be applied. The chosen method has been Givens Rotations due to its good numerical properties and high parallelism. We can see that the number of operations have been reduced compared to the UKF due to the propagation of the sigma factors. On the other hand, more square roots are needed because we are using Givens Rotations to compute the inverse. This can be improve using other technics such as CORDIC when the computation of square roots and complex operations is very critical. Compared to the QR-EKF, the filter computes almost the same number of additions, substractions and multiplications but is more costly in terms of divisions and square roots.

Algorithm 14: SR-UKF

Initialization of coeficients

κ such that $n + \kappa = 3$
$beta = 2$ for Gaussian distributions
$10^{-4} \leq \alpha \leq 1$
$\lambda = \alpha^2(n + k) - n$
$W_0^m = \lambda/(n + \lambda)$
$W_i^m = W_i^c = 1/(2(n + \lambda))$
$W_0^c = \lambda/(n + \lambda) + (1 - \alpha^2 + \beta)$

Initial Cholesky Decomposition

$\mathbf{S}_0 = \sqrt{(n + \kappa)P_{-1}}$

Calculate sigma points

$\boldsymbol{\chi} = \begin{pmatrix} \mathbf{x}_{i|i-1} & \mathbf{x}_{i|i-1} + \mathbf{S}_{:,i} & \mathbf{x}_{i|i-1} - \mathbf{S}_{:,i} \end{pmatrix}$
$for\ k\ \epsilon\ \{1, ..., n\}$

Time update equations

Propagate through state transition function

$\boldsymbol{\chi}' = \mathbf{f}(\boldsymbol{\chi})$

Calculate Mean of the states

$\mathbf{x_{i|i-1}} = \sum_{k=0}^{2n+1} W_k^m \boldsymbol{\chi}'_{\mathbf{k}}$

Update the cholesky factors via QR and rank-one cholesky update to incorporate the effect of process noise. (see Alg. 15)

$\mathbf{S}'_i = QR\begin{pmatrix} \sqrt{W_i^c}(\boldsymbol{\chi}'_{1:2k} - \mathbf{x_{i|i-1}}) & Q^{\frac{1}{2}} \end{pmatrix}$

$\mathbf{S}'_{i+1} = cholupdate(\mathbf{S}'_i, \boldsymbol{\chi}'_0 - \mathbf{x_{i|i-1}}, W_0^c)$

Propagate through observation function

$\boldsymbol{\zeta}' = \mathbf{h}(\boldsymbol{\chi})$

Calculate Mean of the predicted measurements

15 $\mathbf{y'_i} = \sum_{k=0}^{2n+1} W_k^m \boldsymbol{\zeta'_k}$

Update the cholesky factors via QR and rank-one cholesky update to incorporate the effect of measurement noise.

16 $\mathbf{S}'_{y,i} = QR\left(\sqrt{W_i^c}(\boldsymbol{\zeta}'_{1:2k} - \mathbf{y'_i}) \quad R^{\frac{1}{2}}\right)$

17 $\mathbf{S}'_{y,i} = cholupdate(\mathbf{S}'_{y,i}, \boldsymbol{\zeta}'_0 - \mathbf{y'_i}, W_0^c)$

Measurement update equations

18 $\mathbf{P_{xz,i}} = \sum_{k=0}^{2n+1} W_k^c(\boldsymbol{\chi'_k} - \mathbf{x_{i|i\text{-}1}})(\boldsymbol{\zeta'_k} - \mathbf{y'_i})^{\mathbf{H}}$

19 $\mathbf{K_i} = \mathbf{P_{xz,i}}/\mathbf{S}'_{y,i}/\mathbf{S}'^{H}_{y,i}$ where / denotes backward substitution

20 $\mathbf{x_{i|i}} = \mathbf{x_{i|i\text{-}1}} + \mathbf{K_i}(\mathbf{y_i} - \mathbf{y_i}\text{'})$

Update the cholesky factors via p rank-one cholesky updates

21 $U = \mathbf{K_i}\mathbf{S}'_{y,i}$

22 $\mathbf{S}_{i+1} = cholupdate(\mathbf{S}'_{i+1}, U, -1)$

Algorithm 15: Rank-one Update

1 $\mathbf{x} = \mathbf{x}'$
2 **for** $k = 1$ ***to*** n **do**
3 $\quad r = \sqrt{\mathbf{L}_{k,k}^2 + sign \cdot \mathbf{x}_k^2}$
4 $\quad c = r/\mathbf{L}_{k,k}$
5 $\quad s = \mathbf{x}_k/\mathbf{L}_{k,k}$
6 $\quad \mathbf{L}_{k,k} = r$
7 $\quad \mathbf{L}_{k,k+1:n} = (\mathbf{L}_{k,k+1:n} + s \cdot sign \cdot \mathbf{x}_{k+1:p})/c$
8 $\quad \mathbf{x}_{k+1:n} = c \cdot \mathbf{x}_{k+1:n} - s \cdot \mathbf{L}_{k,k+1:n}$
9 **end**
10 $\mathbf{L} = \mathbf{L}'$

	SR-UKF
Additions	$\frac{\mathbf{8n^3}}{3} + \frac{\mathbf{5p^3}}{3} + \frac{\mathbf{5n^2p}}{\mathbf{2}} + \mathbf{2np^2} + \frac{7n^2}{2} - \frac{p^2}{2} + \frac{5np}{2} + \frac{11n}{6} + \frac{5p}{6} + 1$
Substractions	$\frac{\mathbf{p^3}}{3} + \frac{\mathbf{n^2p}}{\mathbf{2}} + \frac{9n^2}{2} + \frac{p^2}{2} + \frac{7np}{2} + \frac{3n}{2} + \frac{13p}{6}$
Multiplications	$\frac{\mathbf{10n^3}}{3} + \frac{\mathbf{8p^3}}{3} + \frac{\mathbf{7n^2p}}{\mathbf{2}} + \mathbf{2np^2} + \frac{15n^2}{2} + \frac{5p^2}{2} + \frac{13np}{2} + \frac{25n}{6} + \frac{5p}{6} + 2$
Divisions	$\frac{n^2p}{2} + \frac{5n^2}{2} + \frac{3p^2}{2} + \frac{3np}{2} + \frac{3n}{2} + \frac{p}{2}$
Square Roots	$\frac{n^2}{2} + \frac{p^2}{2} + np + \frac{13n}{2} - \frac{p}{2} + 3$
Others	atan: $2n + 1$
	Initial Cholesky Decomposition

Table 3.4: Number of operations of SR-UKF

3.4 Analysis Complexity

In this section the complexity of all the filters is summarize. Table 3.5 shows the asymptotic cost of the filters. The number in brackets is the number of operations for the example considered, $n = 4$ and $p = 2$. As expected, the most complex algorithm is the LDL-EKF and the SR-UKF reduces the complexity compared with the UKF. Comparing the total number of operations of the EKF and SR-UKF, the difference in number of operations is negligible, the QR-EKF is only 5 % less complex. However, as shown in the previous section, the SR-EKF compute more complex operations.

	Asymptotic cost
QR-EKF	$\sim 6n^3 + 7n^2p + 3np^2 + \frac{5p^3}{3}$ (670)
LDL-EKF	$\sim \frac{41n^3}{3} + \frac{10p^3}{3} + 9n^2p + 11np^2$ (1366)
UKF	$\sim \frac{25n^3}{3} + \frac{5p^3}{3} + 6n^2p + 8np^2$ (867)
SR-UKF	$\sim 6n^3 + \frac{14p^3}{3} + 7n^2p + 4np^2$ (710)

Table 3.5: Comparison of the asymptotic cost of the filters

3.5 Results

In this section the results of the simulations using the model described in Section 3.1 are shown. The results were obtained using MATLAB floating point arithmetic and noise conditions: $\sigma_u^2 = 10^{-4}$, $\sigma_R^2 = 1$ and $\sigma_\beta^2 = 0.1$. The figures show the evolution of the predicted variances of the estimates and the track estimate with every iteration of the filter.

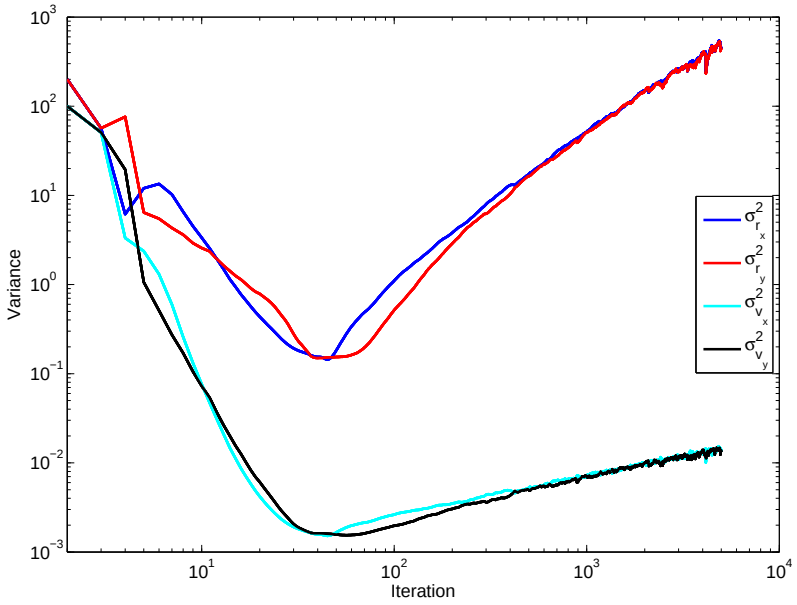


Figure 3.2: Predicted variance of the estimates for the QR-EKF

3.5.1 QR-EKF

The results of the simulations for the QR-EKF are presented in Figures 3.2 and 3.3. The results show that the filter is not stable since the variance of the predicted states is increasing with the number of iterations. This behaviour can also be observe in the prediction of the track, where the variance of the prediction around the ideal track is increasing with every iteration. The instability of the filter is due to the poor linearization perform by the filter, which introduce and accumulates errors in every iteration and make the filter diverge. The application of different QR Decomposition methods to compute the matrix inverse in the Kalman Gain equation has shown no difference since, as shown in Chapter 2, when the condition number of the matrix to inverse is very low, there is no significant difference in the accuracy of all the methods.

3.5.2 LDL-EKF

The results of the simulations for the LDL-EKF are presented in Figures 3.4 and 3.5. The figures show that the performance of the LDL-EKF is exactly the same as in the case of the QR-EKF, the filter is also unstable and the use of square root factors does not improve the performance. This behaviour is

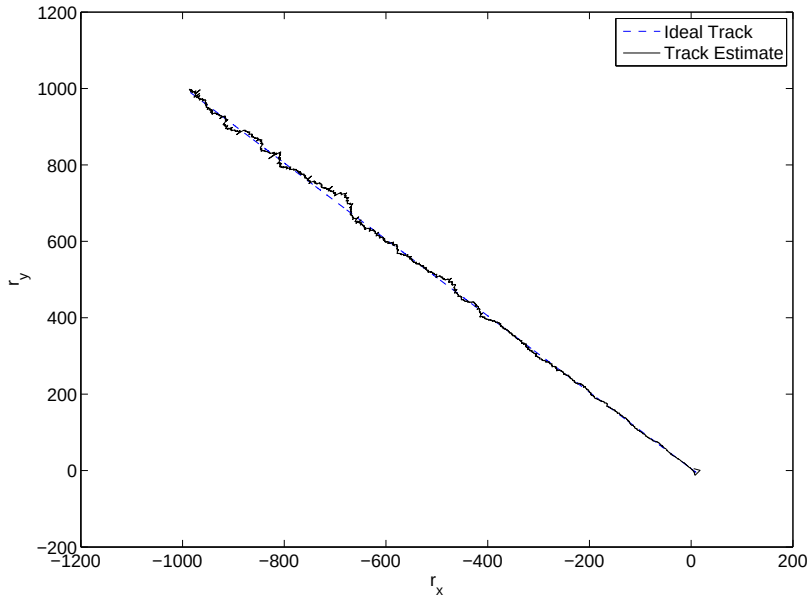


Figure 3.3: Track estimate for the QR-EKF

caused by the fact that the linearization errors are limiting the performance of the filter and all the other errors are negligible.

3.5.3 UKF

The results of the simulations for the UKF are presented in Figures 3.6 and 3.7. In that case, the third order approximation perform by the unscented transform make the variance of the states stable. We can also see that the track is more accurate than in the case of the Extended Kalman Filters.

However, looking in detailed the estimate of the track we can see that the difference between the ideal track and the estimate is increasing. This behaviour is more clear looking at the absolute error of the estimate. In Figure 3.8 we can see that at the beginning the error is very large due to the initial state is not accurate, after a few iteration the error decreases when the filter founds the track but then, instead of converge to a fixed value, the error starts to increase again. We can also see comparing Figures 3.6 and 3.8 that the instant when the error starts to increase coincide with a little increase in the covariance of the position. The reason of this divergence is deeply studied in Section 3.6.

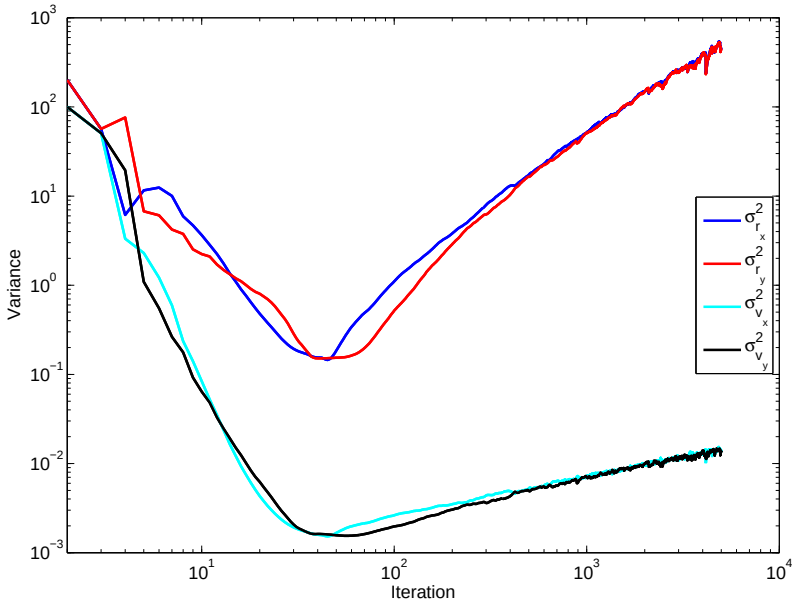


Figure 3.4: Predicted variance of the estimates for the LDL-EKF

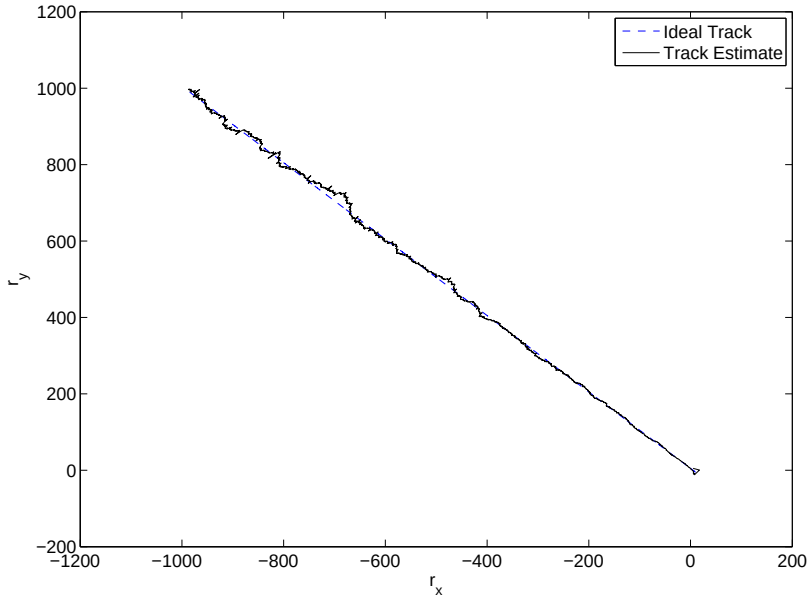


Figure 3.5: Track estimate for the LDL-EKF

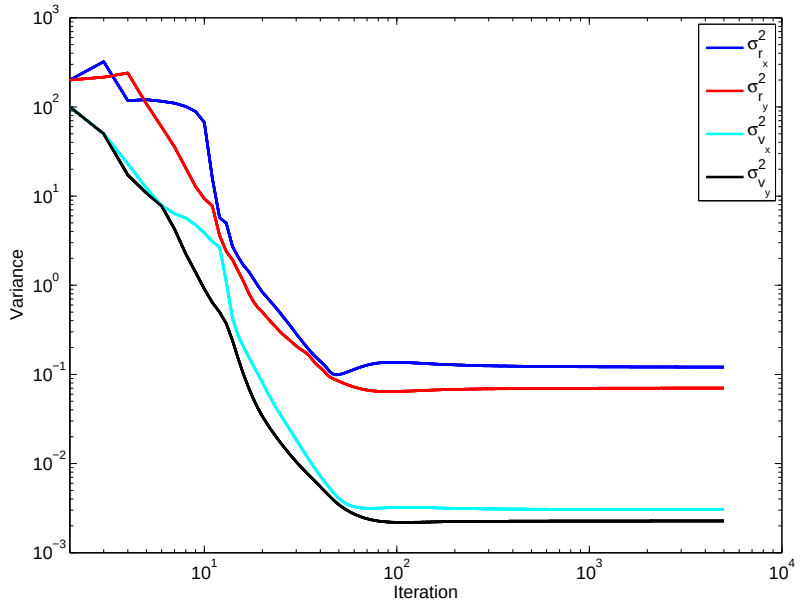


Figure 3.6: Predicted variance of the estimates for the UKF

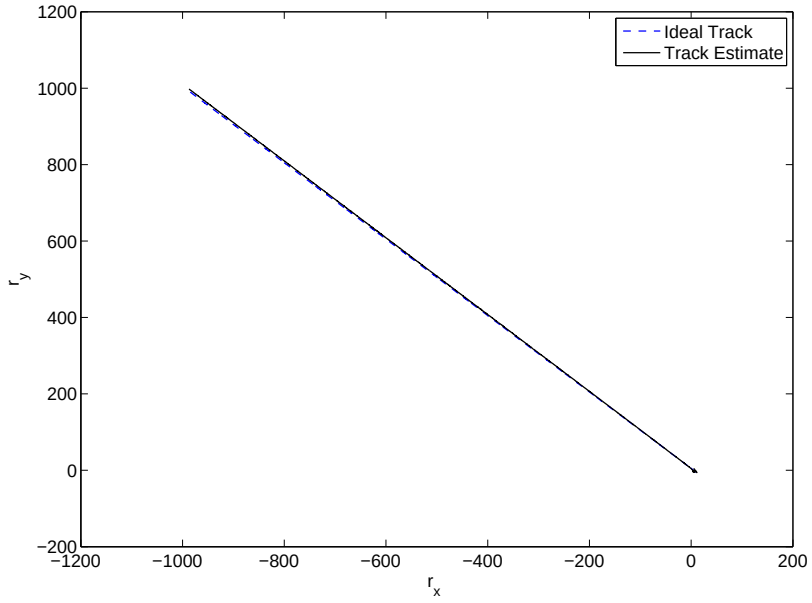


Figure 3.7: Track estimate for the UKF

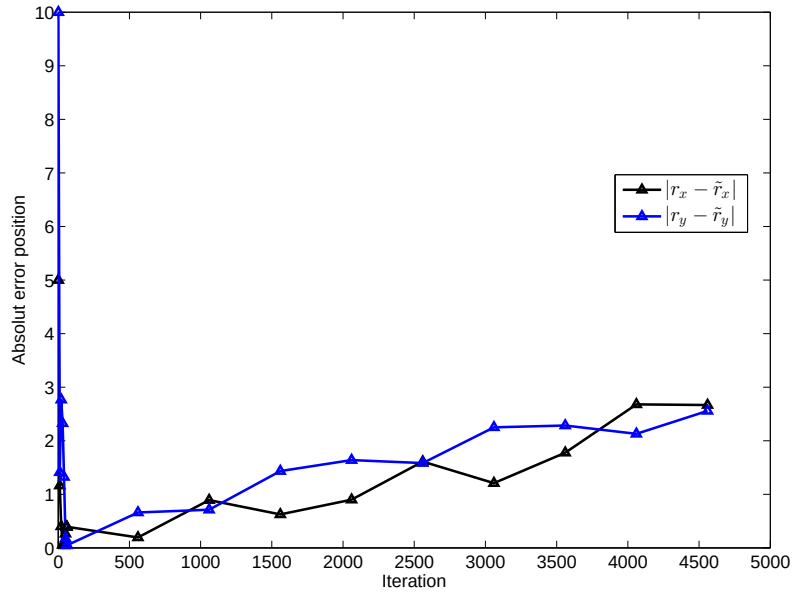


Figure 3.8: Divergence of the estimate of the track using UKF

3.5.4 SR-UKF

The results of the simulations for the SR-UKF are presented in Figures 3.9 and 3.10. As in the case of the UKF, the filter is accurate, the covariance of the states is stable but the same divergence problems are observe. The reduction in the complexity of the filter does not show any drawback in terms of performance.

3.6 Study divergence of UKF

The problem of divergence in the UKF that have been observed in the previous sections was deeply studied in [10]. In the paper, it is proved that the divergence is caused by the rapid state covariance reduction. The state covariance reduction occur overly fast compared with the accuracy of the estimates. That caused the actual value to fall outside the confidence area given by the estimate of the covariance. In the cited paper, the divergence was studied in the case of the estimation of a fixed value. In the system considered here as example, that consists in the tracking of a vehicle, the same divergence due to the rapid decrease in the covariance is observed. Figure 3.11 show how after 15 iterations the rapid reduction of the state covariance

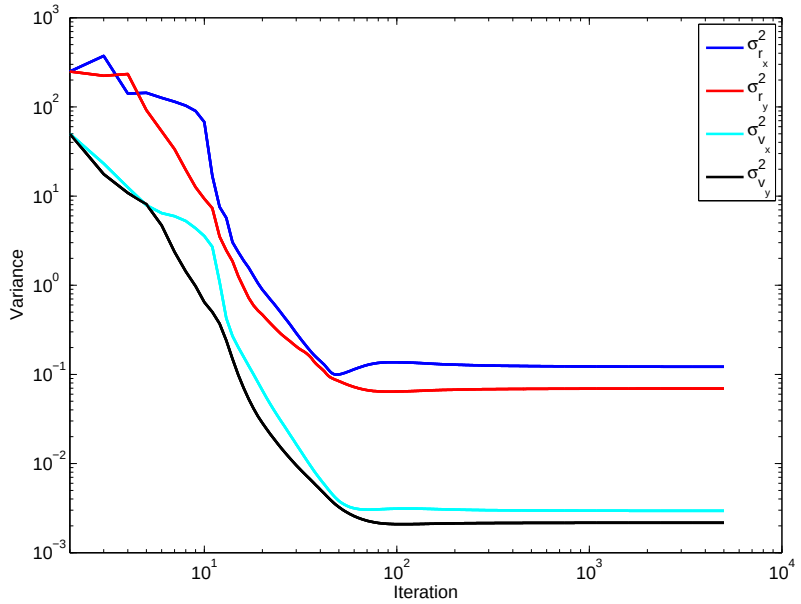


Figure 3.9: Predicted variance of the estimates for the SR-UKF

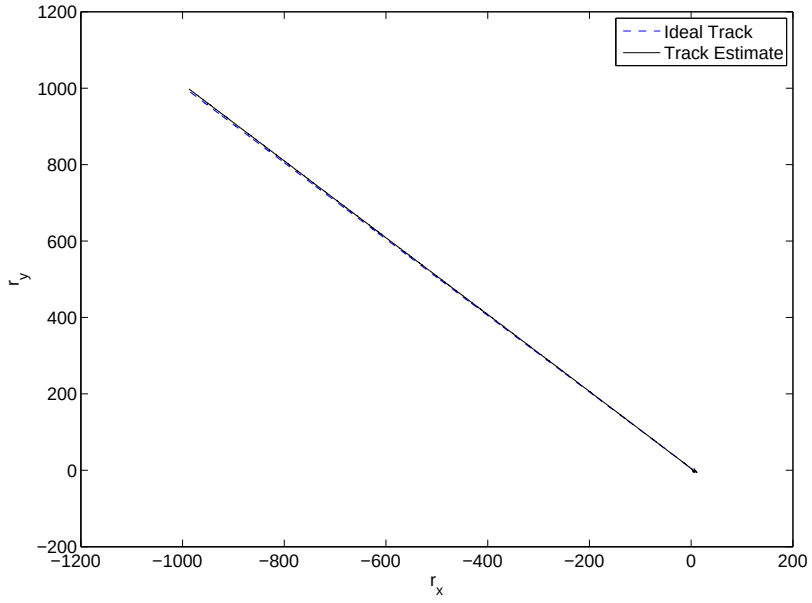


Figure 3.10: Track estimate for the SR-UKF

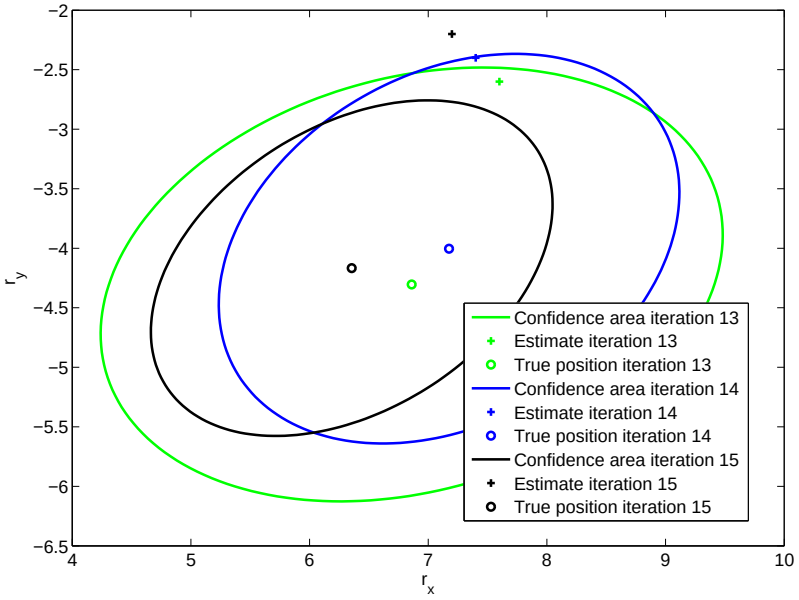


Figure 3.11: Confidence area of state estimate at iteration 13, 14 and 15

makes the actual value to fall outside the ellipsoid that approximates the confidence area P. In Figure 3.12 we can observe that, in the case of the measurements variable z, the actual value falls outside the confidence area after 18 iterations. That makes the filter misinterpret and underweight the new information that comes from the measurements and the estimates begin to move away from the ideal track. The figures have been obtained for the UKF.

In [10], two modifications are proposed based in the idea that the divergence is caused by the over-reduction of the covariance matrix: the first modification is to increase the measurement noise level and the second to use more information from new measurements in the residuals. The second modification has shown to have no improvement in the simulation scenario considered here. The first modification, however, increases the confidence area of the estimates. Figures 3.13 and 3.14 show how increasing artificially the covariance of the measurements the actual values fall inside the confidence area. That avoids the underestimation of the new measurements by the filter and the divergence of the estimate. In Figure 3.15 is observed how with this modification the absolute error is decreasing with every iteration. However, in the cited paper it is shown that with this modification the divergence is only delay. Doing a longer simulation, we can observe that the

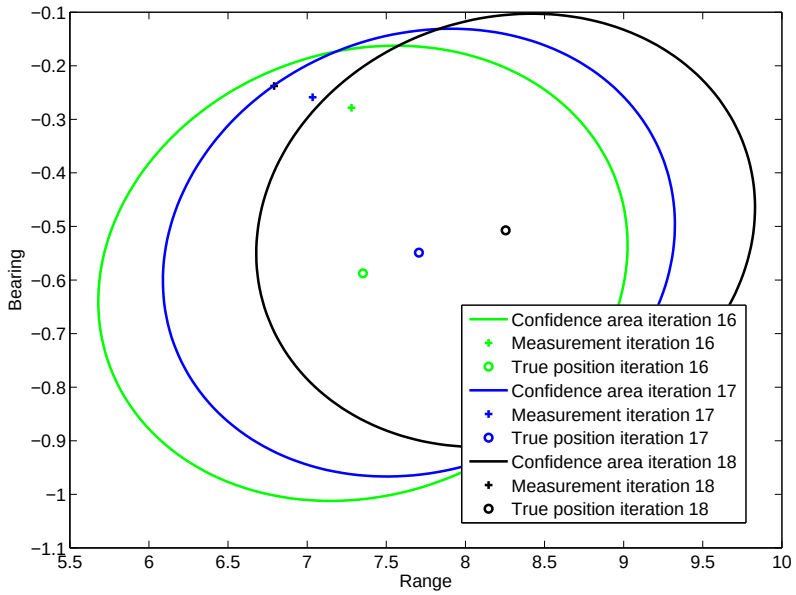


Figure 3.12: Confidence area of measurements at iteration 16, 17 and 18

filter diverges after $1.5 \cdot 10^5$ iterations.

3.7 Conclusions

In this second part of the thesis, an analysis of different Kalman Filter implementations for non-linear models have been done. The filters analyzed have been the EKF, the UKF and its square root forms. The results have shown that the less complex filter is the EKF. However, the complexity of the UKF and SR-UKF is only slightly bigger than in the EKF and clearly outperform it in terms of stability and accuracy. Divergence problems have also been observed in the unscented filters due to the rapid reduction of the covariance matrices that makes the filter underweight the measurements. The modification proposed is to increase artificially the measurement noise. That modification delays the divergence of the filter, but does not avoid it. A further analysis of the divergence is needed to assure completely the stability.

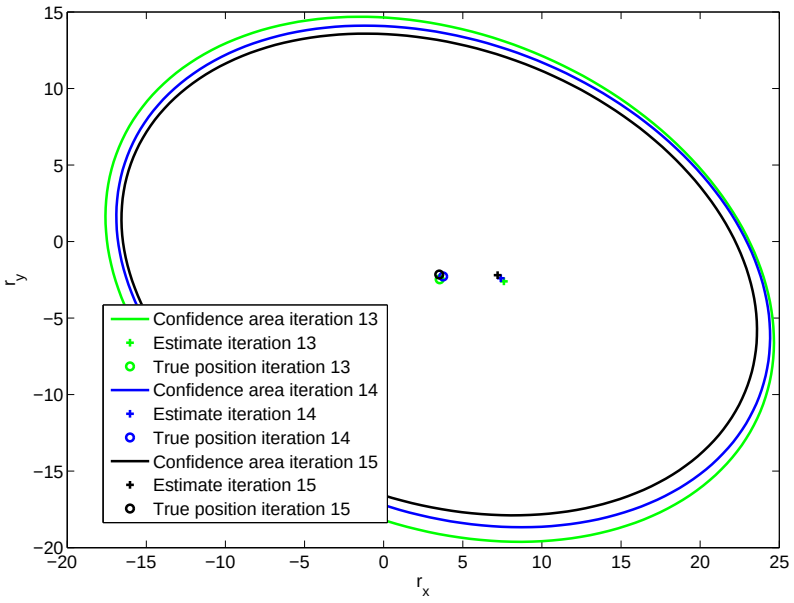


Figure 3.13: Confidence area of state estimate at iteration 13, 14 and 15 using the modified UKF

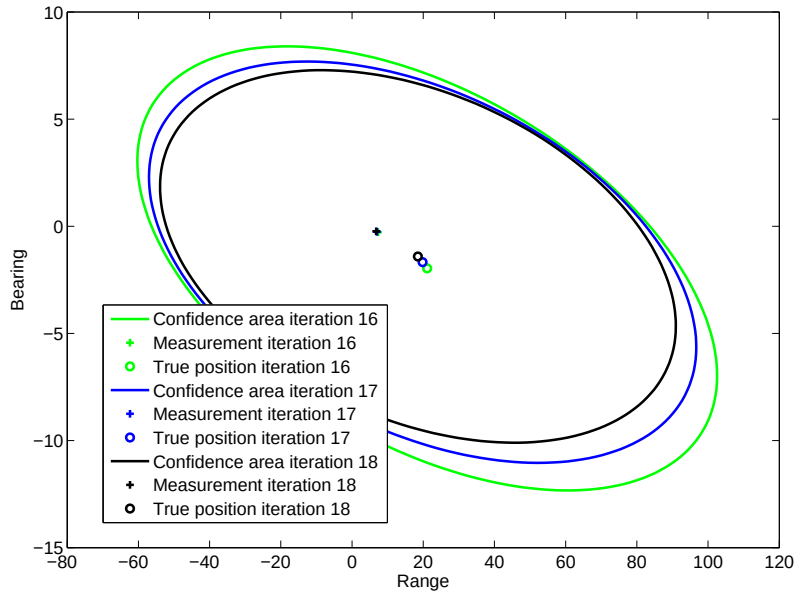


Figure 3.14: Confidence area of measurements at iteration 16, 17 and 18 using the modified UKF

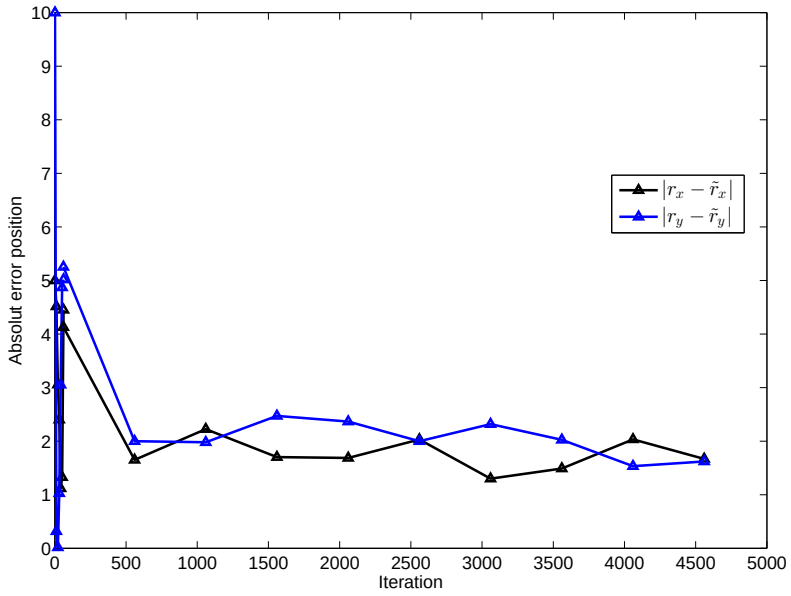


Figure 3.15: Divergence of track estimate using the modified UKF

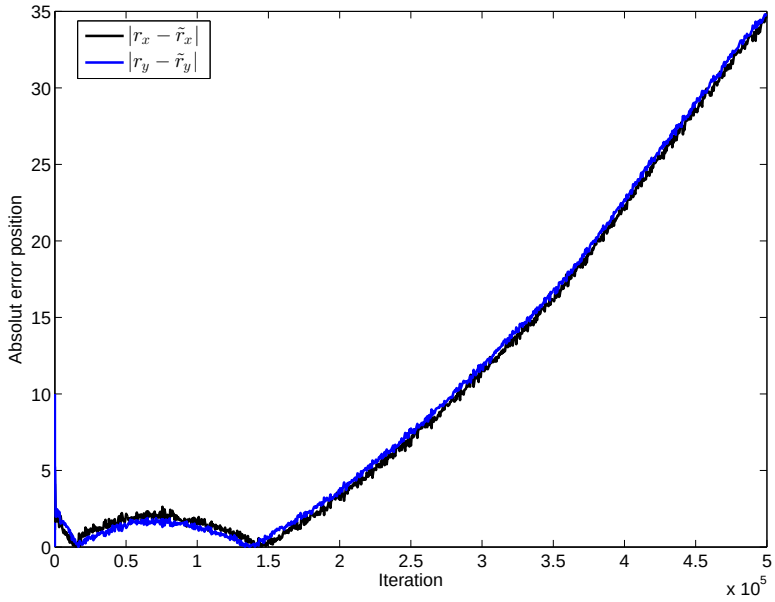


Figure 3.16: Divergence of track estimate using the modified UKF (longer simulation)

Chapter 4

Conclusions

The objective of this thesis was to analyze the impact and improvement in the use of numerical methods and decompositions to the implementation of signal processing algorithms. More specifically, the first part of the thesis focused in solving the problem of the computation of matrix inversion using QR Decomposition and the second part focused in an analysis of different Kalman Filter implementations for non-linear models using different decompositions and numerical methods to improve stability and reduce complexity.

In the QR Decomposition analysis, a new method called Modified Gaussian Elimination was proposed. This new method is less complex and more accurate than all the previous methods developed until now and also have been proved to keep good numerical properties when applied to real systems. The use of MGE results in the computation of a $\mathbf{Q}$ matrix that is not orthogonal, which is not required in the computation of matrix inversion. On the other hand, some applications may required matrix $\mathbf{Q}$ to be orthogonal. In that case, the analysis showed that the best method is Givens Rotations due to its stability, accuracy and low complexity when implemented using parallel structures.

In the second part of the thesis, the performance of the Extended Kalman Filter, the Unscented Kalman Filter and its square root forms have been studied in terms of complexity, accuracy and stability when the system model is non-linear. The different filters have been tested simulating the track of a vehicle with constant velocity. The results have shown that the less complex method is the EKF but at the expense of a high instability due to the poor approximation of the non-linear function. On the other hand, the unscented filters show good stability increasing just slightly the complexity. Among the unscented filters, the SR-UKF is less complex without any drawback in the performance. Even though the unscented filters are by far more stable than the EKF, divergence problems have also been observed because of the

rapid reduction of the covariance estimate compared with the accuracy of the estimate. That makes the estimate to fall outside the confidence area given by the covariance and makes the filter move away from the solution. To solve this problem, an artificial increase of the measurement noise have been proposed in the literature. This modification shows a good improvement in the divergence of the filter for short periods of time. However, longer simulations show that the divergence is not avoid, just delayed.

4.1 Future Work

Unscented Kalman Filters have shown its superiority compared to the Extended Kalman Filters. However, the UKF are also approximating the non-linear functions and for highly non-linear models that can also lead to divergence. The future work should focus in solving the divergence problems of the unscented filters for long time estimations. The use of more than one measurement could also improve the divergence problems. A set of filters that are not considered here are the particle filters. Particle filters result in an exact computation of the probability density function. However, its high complexity makes its used infeasible for real time systems. Until the use of particle filters become possible in real time implementations, the UKF is the best recursive bayesian estimator available.

Bibliography

[1] B. Toni, Tina; Tidor, "Schematic representation of the unscented transform," *Figure13.tif. PLOS Computational Biology. 10.1371/journal.pcbi.1002960.g013*, 2013.

[2] G. Golub and C. Van Loan, *Matrix Computations*, ser. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996. [Online]. Available: http://books.google.es/books?id=mlOa7wPX6OYC

[3] P. Singh, C.K. ; Sushma Honnavara Prasad ; Balsara, "VLSI architecture for matrix inversion using modified gram-schmidt based QR decomposition," in VLSI *Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on.*

[4] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, ser. FPGA '98. New York, NY, USA: ACM, 1998, pp. 191–200. [Online]. Available: http://doi.acm.org/10.1145/275107.275139

[5] J. M. J. M. Lei Ma, Kevin Dickson, "QR decomposition-based matrix inversion for high performance embedded MIMO receivers," IEEE *Transactions on Signal Processing*, 2011.

[6] J. E. Volder, "The CORDIC trigonometric computing technique," *Electronic Computers, IRE Transactions on*, vol. EC-8, no. 3, pp. 330–334, Sept 1959.

[7] X. Wang and M. Leeser, "A truly two-dimensional systolic array FPGA implementation of QR decomposition," *ACM Trans. Embed. Comput. Syst.*, vol. 9, no. 1, pp. 3:1–3:17, Oct. 2009. [Online]. Available: http://doi.acm.org/10.1145/1596532.1596535

[8] L. Trefethen and D. Bau, *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997. [Online]. Available: http://books.google.es/books?id=4Mou5YpRD_kC

[9] R. E. Kalman, "A new approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[10] L. Perea, J. How, L. Breger, and P. Elosegui, "Nonlinearity in sensor fusion: Divergence issues in EKF, modified truncated SOF, and UKF," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6514.

[11] R. Van Der Merwe and E. A. Wan, "The square-root unscented Kalman Filter for state and parameter-estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 6. IEEE, 2001, pp. 3461–3464.

[12] S. Kay, *Fundamentals of Statistical Signal Processing: Estimation theory*, ser. Fundamentals of Statistical Signal Processing. Prentice-Hall PTR, 1998, no. v. 1. [Online]. Available: http://books.google.se/books?id=aFwESQAACAAJ

[13] P. Kaminski, A. E. Bryson, and S. Schmidt, "Discrete square root filtering: A survey of current techniques," *Automatic Control, IEEE Transactions on*, vol. 16, no. 6, pp. 727–736, Dec 1971.

[14] J. Jover and T. Kailath, "A parallel architecture for kalman filter measurement update and parameter estimation," *Automatica*, vol. 22, no. 1, pp. 43–57, 1986.

[15] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman Filter to nonlinear systems," in *Int. symp. aerospace/defense sensing, simul. and controls*, vol. 3, no. 26. Orlando, FL, 1997, pp. 3–2.

Appendix A

Number of operations QR

Method	Additions	Substractions	Multiplications	Divisions	Square Roots	Shifts
HR	$\frac{\mathbf{n^3}}{\mathbf{3}} + \frac{7n^2}{2} - \frac{n}{3}$	$\frac{\mathbf{n^3}}{\mathbf{3}} + \frac{n^2}{2} + \frac{7n}{6}$	$\frac{\mathbf{2n^3}}{\mathbf{3}} + 3n^2 + \frac{13n}{3}$	n	n	0
MGS	$\frac{\mathbf{n^3}}{\mathbf{2}} - \frac{n}{2}$	$\frac{\mathbf{n^3}}{\mathbf{2}} - \frac{n^2}{2}$	$\mathbf{n^3} - n^2$	n^2	n	0
GR	$\frac{\mathbf{5n^3}}{\mathbf{3}} - \frac{n^2}{2} - \frac{7n}{6}$	0	$\frac{\mathbf{10n^3}}{\mathbf{3}} - n^2 - \frac{7n}{3}$	$n^2 - n$	$\frac{n^2}{2} - \frac{n}{2}$	0
CORDIC	$\mathbf{2n^3p} - 2n^2p + n^2 - n$	0	0	0	0	$\mathbf{3n^3p} - 3n^2p + \frac{3n^2}{2} - \frac{3n}{2}$
MSGR	$\frac{\mathbf{5n^3}}{\mathbf{6}} - \frac{n^2}{2} - \frac{n}{3}$	$\frac{\mathbf{5n^3}}{\mathbf{6}} - \frac{n^2}{2} - \frac{n}{3}$	$\frac{\mathbf{5n^3}}{\mathbf{6}} + \frac{n^2}{2} - \frac{n}{6}$	$n^2 - n$	0	0
MGE	0	$\frac{\mathbf{5n^3}}{\mathbf{6}} - \frac{n^2}{2} - \frac{n}{3}$	$\frac{\mathbf{5n^3}}{\mathbf{6}} - \frac{n^2}{2} - \frac{n}{3}$	$\frac{n^2}{2} - \frac{n}{2}$	0	0

Table A.1: Number of operations to compute QR decomposition