Implementing the Matrix Exponential Function on Embedded Processors

James Lebak
Andrea Wadell
Massachusetts Institute of Technology
Lincoln Laboratory

Eighth Annual High-Performance Embedded Computing Workshop (HPEC 2004)
30 Sep 2004

This work is sponsored by the United States Navy under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Definition
[Moler and Van Loan,2003]

The solution to the differential equation

$$\dot{x} = Ax(t)$$

$$x(0) = x_0$$

is given by

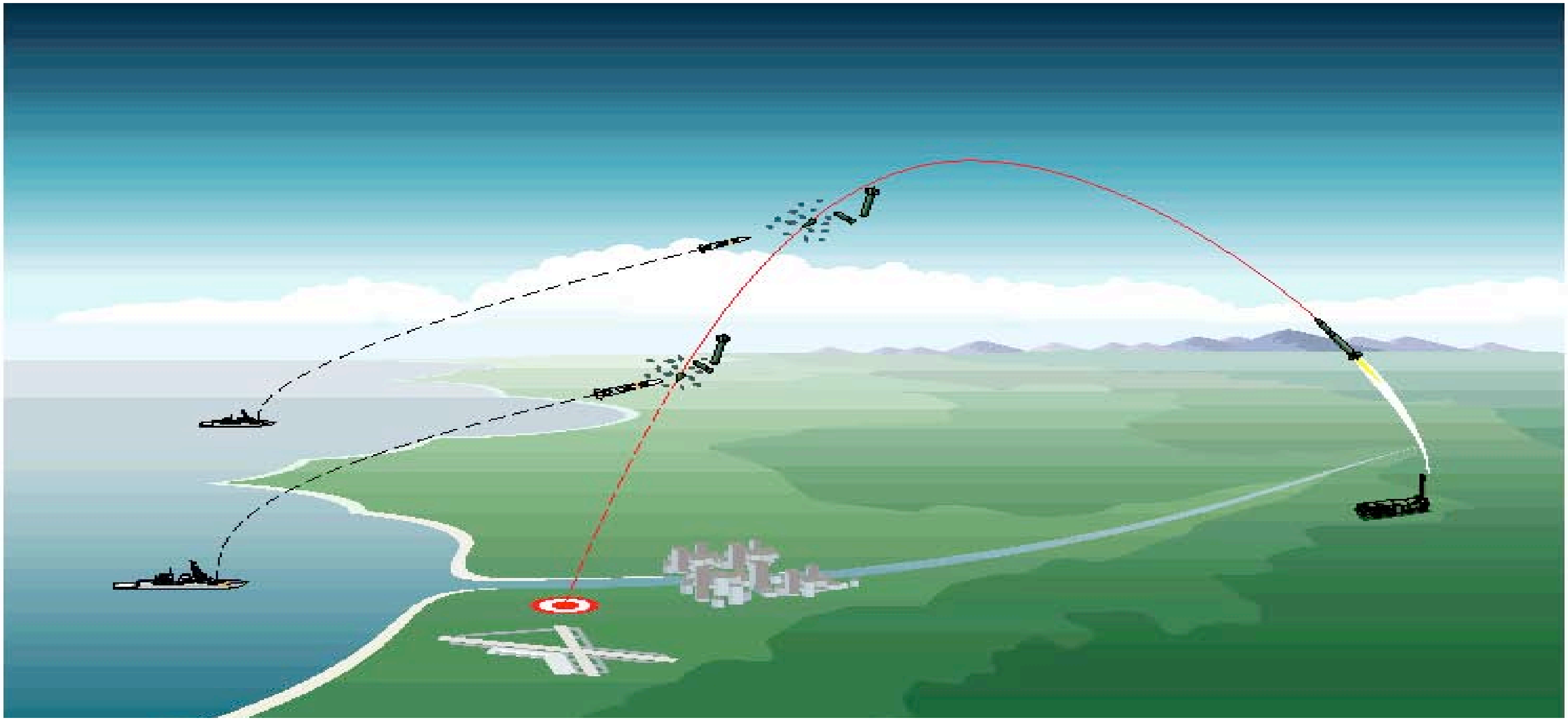
$$x(t) = e^{At} x_0$$

Where e^{At} **is the matrix exponential function,**

$$e^{At} = I + At + \frac{A^2 t^2}{2!} + \dots$$

Notice that if $A = [a_{ij}], e^{At} \neq [e^{a_{ij} t}]$ **in general.**

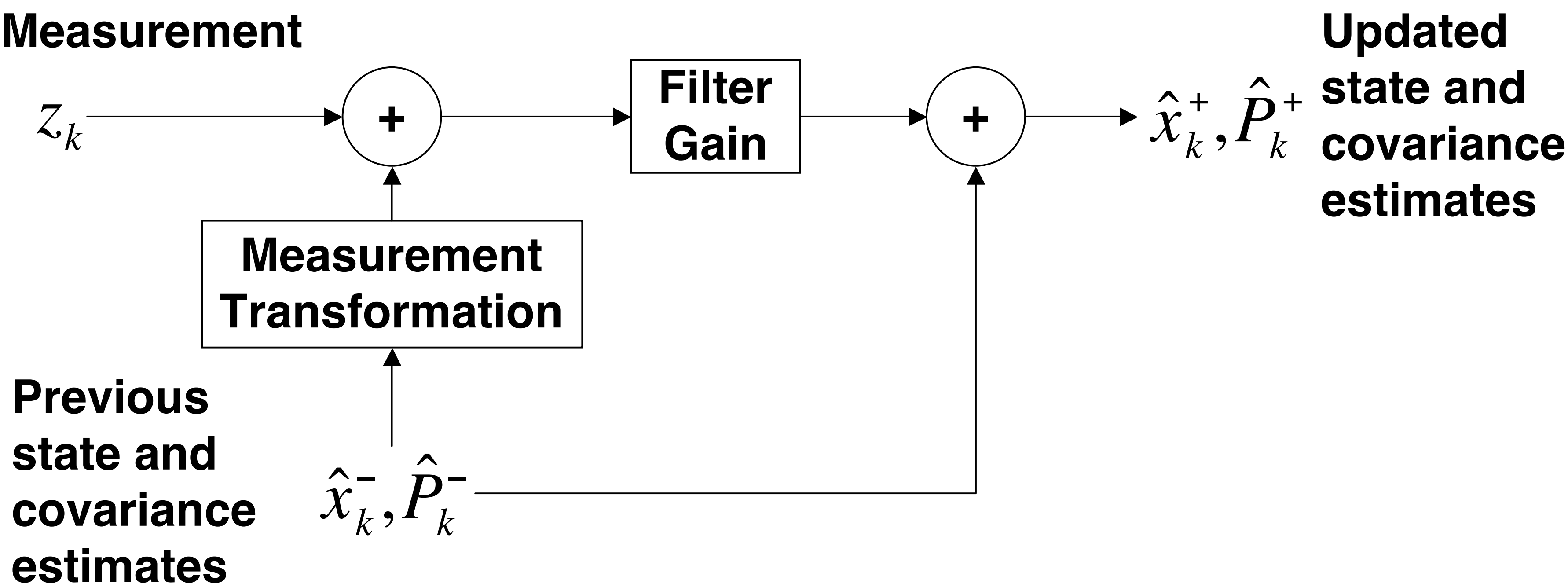
Application: Ballistic Target Tracking



- **Tracking of a ballistic target using noisy measurements**
- **Tracking accomplished using the *extended Kalman filter***
  - **"extended" means that system dynamics are non-linear**

The Extended Kalman Filter

Estimate next state based on previous state and new measurement



$$\hat{P}_k^+ = \Phi \hat{P}_k^- \Phi + Q(t),$$

where $\Phi = e^{Jt}$ **for a matrix** J,

and $Q(t)$ **is the process noise covariance.**

Calculation Overview

Preferred method, Padé approximation, is only valid when ||A|| is small

Use the fact that $e^{A} = (e^{A/m})^{m}$

1. **Choose an integer** j **and scale** A **by** $m=2^{j}$
2. **Use a Padé approximation to calculate** $E = e^{A/2^{j}}$
3. **Perform** j **matrix multiplies to calculate** $E^{2^{j}}$

This technique is referred to as "scaling and squaring" [4,5].

Padé Iteration Algorithm

```
X = A;
c = 1;
E = I;
D = I;
for(k = 1; k <= q; k++) // q=number of iterations
{
   c = c * (q-k+1) / (k*(2*q-k+1));
   X = A*X;                    // Matrix multiply
   E = E + cX;                 // Matrix scale and add
   if (k is even)              // Matrix add or subtract
     D = D + cX;
   else
     D = D - cX;
}
E = D\E;                       // Solve using LU factorization
```

Implementation Overview

Step	Operations	Percentage of op count
Scale the matrix A	Elementwise multiply	<2%
Padé iteration	Matrix multiply, scale, add	50-75%
LU and backsolve		3-6%
Repeated squaring	Matrix multiply	13-50%

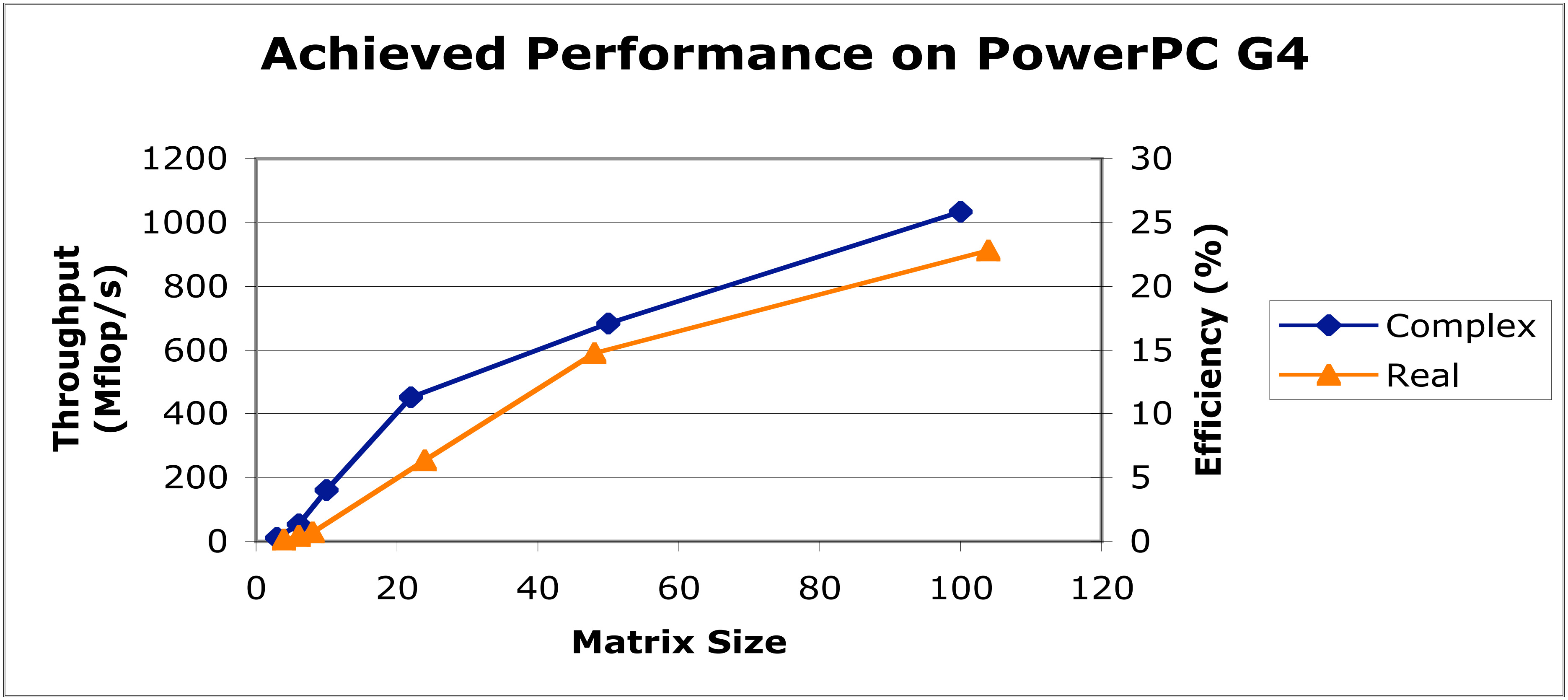
Op counts assume 6 Padé iterations

Implementation Features

- Single-precision real or complex float
- C++
- Uses an object for storage
- Calls VSIPL routines
- Uses Altivec-optimized matrix multiply
- Choose accuracy to match limits of single-precision calculations

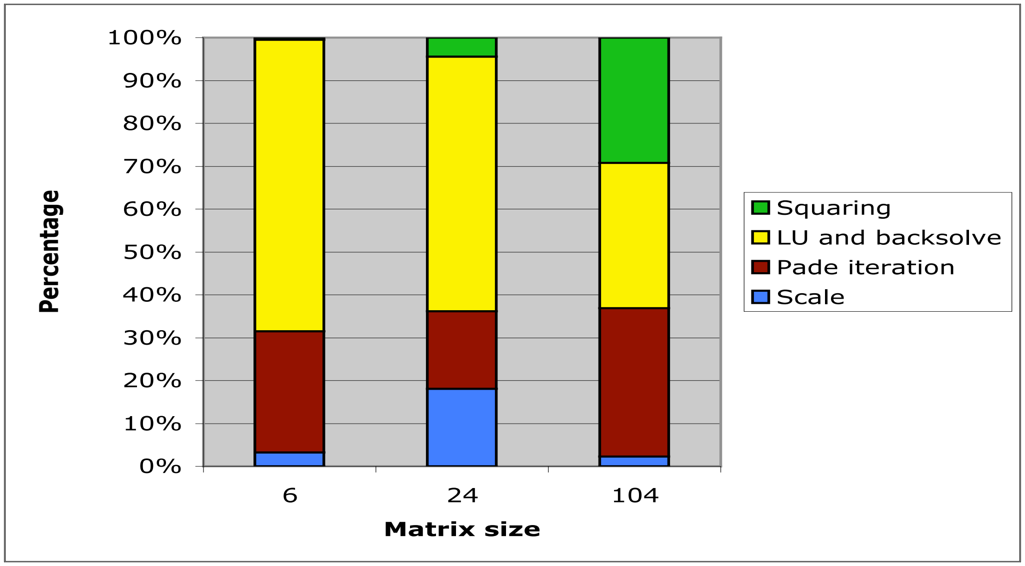
```
void create(Matrix<T> &A,          // Allocates memory & initializes
            Matrix<T> &E);         // LU factorization
void run(   Matrix<T> &A,          // Performs computation
            Matrix<T> &E);
```

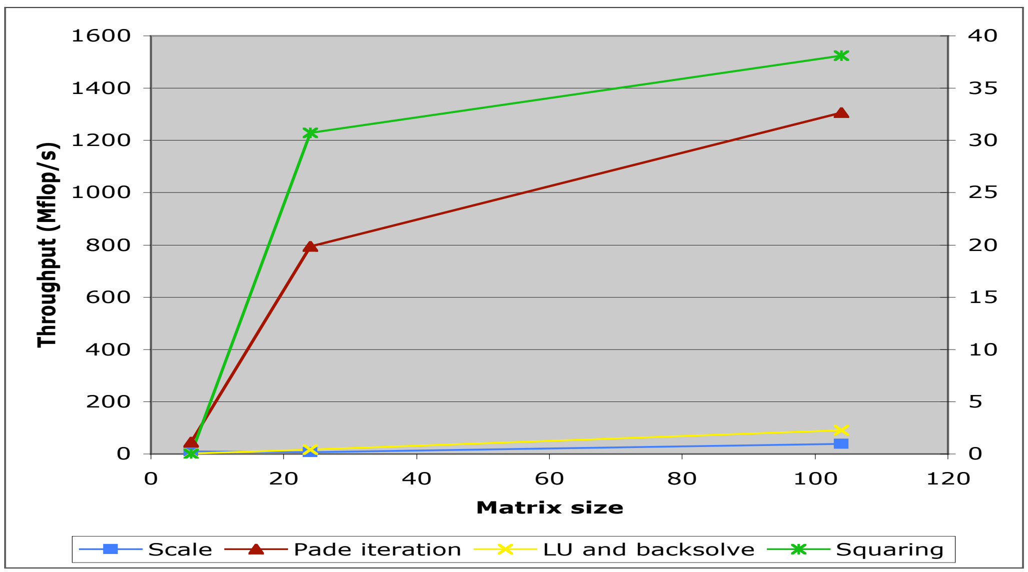
Performance



- **Platform: Mercury 500 MHz PowerPC G4**
- **Achieves respectable performance for large matrices**
- **For tracking, sizes of interest are small — 6x6 matrices**
  - **A tuned implementation could be produced for this size**

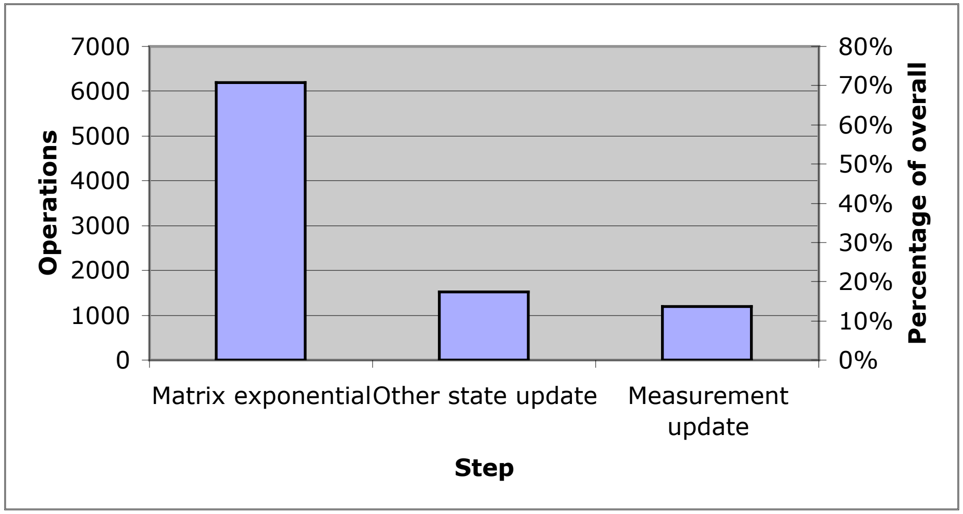
Performance Breakdown

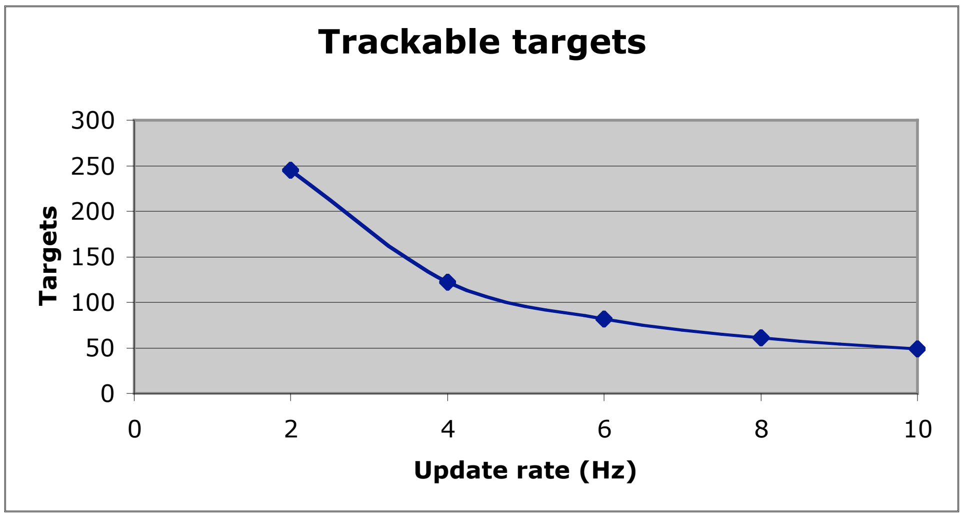




- Performance breakdown on PowerPC G4
- Steps based on matrix multiply are more efficient than other steps
- For large matrices, matrix multiply steps still consume most of the execution time
- LU/backsolve is a substantial percentage of time despite being a low percentage of the op count

The Matrix Exponential in Tracking





- **Matrix exponential is a substantial part of the EKF's operation count**
- **How many targets could a single processor track?**
  - **Assume 500 MHz PPC G4**
  - **Use execution time of 6x6 real matrix exponential**
  - **Assume remainder of EKF has efficiency comparable to LU factorization (~0.04%)**
  - **Vary track rate from 2-10 Hz**
- **A single processor can potentially track many targets**

Conclusions

- Matrix exponential function is important for tracking applications
- A large percentage of the operations are matrix multiply functions
- An efficient implementation of this function allows it to be used in an extended Kalman filter
- Many targets can be tracked using even a single processor
  - Using multiple processors obviously allows more targets to be tracked

References

[1] Michael Athans, Robert H. Whiting, and Michael Gruber. A suboptimal estimation algorithm with probabilistic editing for false measurements with applications to target tracking with wake phenomena. *IEEE Transactions on Automatic Control*, 22(3):372–384, June 1977.

[2] Y. Bar-Shalom, X. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley and Sons, 2001.

[3] A. Farina, B. Ristic, and D. Benvenuti. Tracking a ballistic target: comparison of several nonlinear filters. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):854–867, July 2002.

[4] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[5] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49,March 2003.

[6] B. Ristic, A. Farina, D. Benvenuti and M. S. Arulampalam. Performance bounds and comparison of nonlinear filters for tracking a ballistic object on re-entry. *IEE Proceedings on Radar and Sonar Navigation*, 150(2):65–70, April 2003.