# Fundamentals of Kalman Filtering: A Practical Approach

Paul Zarchan

# Seminar Outline - 1

- Numerical Techniques
  - Required background
  - Introduction to source code
- Method of Least Squares
  - How to build batch process least squares filter
  - Performance and software comparison of different order filters
- Recursive Least Squares Filtering
  - How to make batch process filter recursive
  - Formulas and properties of various order filters

# Seminar Outline - 2

- Polynomial Kalman Filters
  - Relationship to recursive least squares filter
  - How to apply Kalman filtering and Riccati equations
  - Examples of utility in absence of a priori information
- Kalman Filters in a Non Polynomial World
  - How polynomial filter performs when mismatched to real world
  - Improving Kalman filter with a priori information
- Continuous Polynomial Kalman Filter
  - How continuous filters can be used to understand discrete filters
  - Using transfer functions to represent and understand Kalman filters

# Seminar Outline - 3

- Extended Kalman Filtering
  - How to apply equations to a practical example
  - Showing what can go wrong with several different design approaches
- Drag and Falling Object
  - Designing two different extended filters for this problem
  - Demonstrating the importance of process noise
- Cannon Launched Projectile Tracking Problem
  - Comparing Cartesian and polar extended filters in terms of performance and software requirements
  - Comparing extended and linear Kalman filters in terms of performance and robustness

# Seminar Outline - 4

- Tracking a Since Wave
  - Developing different filter formulations and comparing results
- Satellite Navigation (Simplified GPS Example)
  - Step by step approach for determining receiver location based on satellite range measurements
- Biases
  - Filtering techniques for estimating biases in GPS example
- Linearized Kalman Filtering
  - Two examples and comparisons with extended filter
- Miscellaneous Topics
  - Detecting filter divergence
  - Practical illustration of inertial aiding

# Seminar Outline - 5

- Tracking an Exoatmospheric Target
  - Comparison of Kalman and Fading Memory Filters
- Miscellaneous Topics 2
  - Using additional measurements
  - Batch processing
  - Making filters adaptive
- Filter Banks
- Practical Uses of Chain Rule
  - 3D GPS
- Finite Memory Filter
- Extra
  - Stereo, Filtering Options, Cramer-Rao & More

# Numerical Basics

# Numerical Basics
# Overview

- Simple vector and matrix operations
- Numerical integration
- Noise and random variables
    - Definitions
    - Gaussian noise example
    - Simulating white noise
- State space notation
- Fundamental matrix

# Simple Vector and Matrix Operations

# Vector Operations - 1

## Column vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{bmatrix}$$

## Example of vector

$$\mathbf{r} = \begin{bmatrix} 5 \\ 7 \\ 2 \end{bmatrix}$$

## Vector addition

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ x_3 + y_3 \\ . \\ . \\ . \\ x_n + y_n \end{bmatrix}$$

# Vector Operations - 2

## Vector subtraction

$$\mathbf{x} - \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \\ x_3 - y_3 \\ . \\ . \\ . \\ x_n - y_n \end{bmatrix}$$

## Example

$$\mathbf{r} = \begin{bmatrix} 5 \\ 7 \\ 2 \end{bmatrix} \qquad \mathbf{s} = \begin{bmatrix} -2 \\ 6 \\ 4 \end{bmatrix}$$

$$\mathbf{r} + \mathbf{s} = \begin{bmatrix} 5 \\ 7 \\ 2 \end{bmatrix} + \begin{bmatrix} -2 \\ 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 13 \\ 6 \end{bmatrix}$$

$$\mathbf{r} - \mathbf{s} = \begin{bmatrix} 5 \\ 7 \\ 2 \end{bmatrix} - \begin{bmatrix} -2 \\ 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 7 \\ 1 \\ -2 \end{bmatrix}$$

# Vector Operations - 3

## Column vector transpose

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{bmatrix} \longrightarrow \mathbf{x}^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{bmatrix}^T = \begin{bmatrix} x_1 & x_2 & x_3 & . & . & . & x_n \end{bmatrix}$$

## Row vector transpose

$$\mathbf{z} = \begin{bmatrix} z_1 & z_2 & z_3 & . & . & . & z_n \end{bmatrix} \longrightarrow \mathbf{z}^T = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ . \\ . \\ . \\ z_n \end{bmatrix}$$

## Numerical example

$$\mathbf{r} = \begin{bmatrix} 5 \\ 7 \\ 2 \end{bmatrix} \longrightarrow \mathbf{r}^T = \begin{bmatrix} 5 & 7 & 2 \end{bmatrix}$$

# Simple Matrix Operations - 1

## Matrix is an array of elements

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & . & . & . & a_{1n} \\ a_{21} & a_{22} & a_{23} & . & . & . & a_{2n} \\ a_{31} & a_{32} & a_{33} & . & . & . & a_{3n} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ a_{m1} & a_{m2} & a_{m3} & . & . & . & a_{mn} \end{bmatrix}$$

## Example of 3 by 3 square matrix

$$\mathbf{R} = \begin{bmatrix} -1 & 6 & 2 \\ 3 & 4 & -5 \\ 7 & 2 & 8 \end{bmatrix}$$

**Diagonal elements are -1, 4 and 8**

## Matrix addition only defined when matrices have same dimensions

$$\mathbf{S} = \begin{bmatrix} 2 & -6 \\ 1 & 5 \\ -2 & 3 \end{bmatrix} \qquad \mathbf{T} = \begin{bmatrix} 9 & 1 \\ -7 & 2 \\ 5 & 8 \end{bmatrix} \longrightarrow \mathbf{S} + \mathbf{T} = \begin{bmatrix} 2 & -6 \\ 1 & 5 \\ -2 & 3 \end{bmatrix} + \begin{bmatrix} 9 & 1 \\ -7 & 2 \\ 5 & 8 \end{bmatrix} = \begin{bmatrix} 11 & -5 \\ -6 & 7 \\ 3 & 11 \end{bmatrix}$$

# FORTRAN Program To Perform Matrix Addition

```
          IMPLICIT REAL*8 (A-H)
          IMPLICIT REAL*8 (O-Z)
          REAL*8 S(3,2),T(3,2),U(3,2)
          S(1,1)=2.
          S(1,2)=-6.
          S(2,1)=1.
          S(2,2)=5.
          S(3,1)=-2.
          S(3,2)=3.
          T(1,1)=9.
          T(1,2)=1.
          T(2,1)=-7.
          T(2,2)=2.
          T(3,1)=5.
          T(3,2)=8.
          CALL MATADD(S,3,2,T,U)
          WRITE(9,*)U(1,1),U(1,2)
          WRITE(9,*)U(2,1),U(2,2)
          WRITE(9,*)U(3,1),U(3,2)
          PAUSE
          END

          SUBROUTINE MATADD(A,IROW,ICOL,B,C)
          IMPLICIT REAL*8 (A-H)
          IMPLICIT REAL*8 (O-Z)
          REAL*8 A(IROW,ICOL),B(IROW,ICOL),C(IROW,ICOL)
          DO 120 I=1,IROW
          DO 120 J=1,ICOL
                    C(I,J)=A(I,J)+B(I,J)
120       CONTINUE
          RETURN
          END
```

# MATLAB and True BASIC Equivalents For Performing Matrix Addition

## MATLAB

```
S=[2 -6;1 5;-2 3];
T=[9 1;-7 2;5 8];
U=S+T
```

## True BASIC

```
OPTION NOLET
DIM S(3,2),T(3,2),U(3,2)
S(1,1)=2.
S(1,2)=-6.
S(2,1)=1.
S(2,2)=5.
S(3,1)=-2.
S(3,2)=3.
T(1,1)=9.
T(1,2)=1.
T(2,1)=-7.
T(2,2)=2.
T(3,1)=5.
T(3,2)=8.
MAT U=S+T
MAT PRINT U
END
```

# Simple Matrix Operations - 2

## Matrix subtraction

$$\mathbf{S} - \mathbf{T} = \begin{bmatrix} 2 & -6 \\ 1 & 5 \\ -2 & 3 \end{bmatrix} - \begin{bmatrix} 9 & 1 \\ -7 & 2 \\ 5 & 8 \end{bmatrix} = \begin{bmatrix} -7 & -7 \\ 8 & 3 \\ -7 & -5 \end{bmatrix}$$

## Transpose of matrix

$$\mathbf{S}^{T} = \begin{bmatrix} 2 & -6 \\ 1 & 5 \\ -2 & 3 \end{bmatrix}^{T} = \begin{bmatrix} 2 & 1 & -2 \\ -6 & 5 & 3 \end{bmatrix}$$

## Matrix is symmetric if rows and columns can be interchanged or

$$\mathbf{A} = \mathbf{A}^{T}$$

## The following matrix is symmetric

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

## Because

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 9 \end{bmatrix}^{T} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

# FORTRAN Program To Perform Matrix Subtraction

```
      IMPLICIT REAL*8 (A-H)
      IMPLICIT REAL*8 (O-Z)
      REAL*8 S(3,2),T(3,2),U(3,2)
      S(1,1)=2.
      S(1,2)=-6.
      S(2,1)=1.
      S(2,2)=5.
      S(3,1)=-2.
      S(3,2)=3.
      T(1,1)=9.
      T(1,2)=1.
      T(2,1)=-7.
      T(2,2)=2.
      T(3,1)=5.
      T(3,2)=8.
      CALL MATSUB(S,3,2,T,U)
      WRITE(9,*)U(1,1),U(1,2)
      WRITE(9,*)U(2,1),U(2,2)
      WRITE(9,*)U(3,1),U(3,2)
      PAUSE
      END

      SUBROUTINE MATSUB(A,IROW,ICOL,B,C)
      IMPLICIT REAL*8 (A-H)
      IMPLICIT REAL*8 (O-Z)
      REAL*8 A(IROW,ICOL),B(IROW,ICOL),C(IROW,ICOL)
      DO 120 I=1,IROW
      DO 120 J=1,ICOL
            C(I,J)=A(I,J)-B(I,J)
120   CONTINUE
      RETURN
      END
```

# FORTRAN Program To Perform Matrix Transpose

```
      IMPLICIT REAL*8 (A-H)
      IMPLICIT REAL*8 (O-Z)
      REAL*8 S(3,2),ST(2,3)
      S(1,1)=2.
      S(1,2)=-6.
      S(2,1)=1.
      S(2,2)=5.
      S(3,1)=-2.
      S(3,2)=3.
      CALL MATTRN(S,3,2,ST)
      WRITE(9,*)ST(1,1),ST(1,2),ST(1,3)
      WRITE(9,*)ST(2,1),ST(2,2),ST(2,3)
      PAUSE
      END


      SUBROUTINE MATTRN(A,IROW,ICOL,AT)
      IMPLICIT REAL*8 (A-H)
      IMPLICIT REAL*8 (O-Z)
      REAL*8 A(IROW,ICOL),AT(ICOL,IROW)
      DO 105 I=1,IROW
      DO 105 J=1,ICOL
      AT(J,I)=A(I,J)
105   CONTINUE
      RETURN
      END
```

# Simple Matrix Operations - 3

A matrix with m rows and n columns can only be multiplied by a matrix with n rows and q columns

- Multiply each element of the rows of matrix A with each element of the columns of matrix B
- New matrix has m rows and q columns

## Example

$$\mathbf{RS} = \begin{bmatrix} -1 & 6 & 2 \\ 3 & 4 & -5 \\ 7 & 2 & 8 \end{bmatrix} \begin{bmatrix} 2 & -6 \\ 1 & 5 \\ -2 & 3 \end{bmatrix} = \begin{bmatrix} -1*2+6*1+2*(-2) & -1*(-6)+6*5+2*3 \\ 3*2+4*1-5*(-2) & 3*(-6)+4*5-5*3 \\ 7*2+2*1+8*(-2) & 7*(-6)+2*5+8*3 \end{bmatrix} = \begin{bmatrix} 0 & 42 \\ 20 & -13 \\ 0 & -8 \end{bmatrix}$$

# FORTRAN Program To Perform Matrix Multiplication

```
          IMPLICIT REAL*8 (A-H)
          IMPLICIT REAL*8 (O-Z)
          REAL*8 S(3,2),R(3,3),RS(3,2)
          S(1,1)=2.
          S(1,2)=-6.
          S(2,1)=1.
          S(2,2)=5.
          S(3,1)=-2.
          S(3,2)=3.
          R(1,1)=-1.
          R(1,2)=6.
          R(1,3)=2.
          R(2,1)=3.
          R(2,2)=4.
          R(2,3)=-5.
          R(3,1)=7.
          R(3,2)=2.
          R(3,3)=8.
          CALL MATMUL(R,3,3,S,3,2,RS)
          WRITE(9,*)RS(1,1),RS(1,2)
          WRITE(9,*)RS(2,1),RS(2,2)
          WRITE(9,*)RS(3,1),RS(3,2)
          PAUSE
          END

          SUBROUTINE MATMUL(A,IROW,ICOL,B,JROW,JCOL,C)
          IMPLICIT REAL*8 (A-H)
          IMPLICIT REAL*8 (O-Z)
          REAL*8 A(IROW,ICOL),B(JROW,JCOL),C(IROW,JCOL)
          DO 110 I=1,IROW
          DO 110 J=1,JCOL
                  C(I,J)=0.
                  DO 110 K=1,ICOL
                          C(I,J)= C(I,J)+A(I,K)*B(K,J)
 110      CONTINUE
          RETURN
          END
```

# Simple Matrix Operations - 4

Identity matrix has unity diagonal elements and zeroes elsewhere

Two by two

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Three by three

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix times it’s inverse is identity matrix

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

# Matrix Inverse For Two By Two Square Matrix

## Two by two formula

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

## Numerical example

$$\mathbf{A} = \begin{bmatrix} 2 & -4 \\ 1 & 3 \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{2*3-(-4)*1}\begin{bmatrix} 3 & 4 \\ -1 & 2 \end{bmatrix} = \frac{1}{10}\begin{bmatrix} 3 & 4 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} .3 & .4 \\ -.1 & .2 \end{bmatrix}$$

## Check

$$\mathbf{A}^{-1}\mathbf{A} = \begin{bmatrix} .3 & .4 \\ -.1 & .2 \end{bmatrix}\begin{bmatrix} 2 & -4 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} .3*2+.4*1 & .3*(-4)+.4*3 \\ -.1*2+.2*1 & -.1*(-4)+.2*3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

# Matrix Inverse For Three By Three Square Matrix - 1

## Three by three formula

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{aei + bfg + cdh - ceg - bdi - afh} \begin{bmatrix} ei\text{-}fh & ch\text{-}bi & bf\text{-}ec \\ gf\text{-}di & ai\text{-}gc & dc\text{-}af \\ dh\text{-}ge & gb\text{-}ah & ae\text{-}bd \end{bmatrix}$$

## For example, given

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{bmatrix}$$

## Coefficient given by

$$\frac{1}{aei + bfg + cdh - ceg - bdi - afh} = \frac{1}{1*5*10+2*6*7+3*4*8-3*5*7-2*4*10-1*6*8} = \frac{-1}{3}$$

# Matrix Inverse For Three By Three Square Matrix - 2

## Matrix itself given by

$$\begin{bmatrix} ei\text{-}fh & ch\text{-}bi & bf\text{-}ec \\ gf\text{-}di & ai\text{-}gc & dc\text{-}af \\ dh\text{-}ge & gb\text{-}ah & ae\text{-}bd \end{bmatrix} = \begin{bmatrix} 5*10-6*8 & 3*8-2*10 & 2*6-5*3 \\ 7*6-4*10 & 1*10-7*3 & 4*3-1*6 \\ 4*8-7*5 & 7*2-1*8 & 1*5-2*4 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -3 \\ 2 & -11 & 6 \\ -3 & 6 & -3 \end{bmatrix}$$

## Therefore inverse of A computed as

$$\mathbf{A}^{-1} = \frac{-1}{3}\begin{bmatrix} 2 & 4 & -3 \\ 2 & -11 & 6 \\ -3 & 6 & -3 \end{bmatrix} = \begin{bmatrix} -2/3 & -4/3 & 1 \\ -2/3 & 11/3 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

## Check

$$\mathbf{A}^{-1}\mathbf{A} = \begin{bmatrix} -2/3 & -4/3 & 1 \\ -2/3 & 11/3 & -2 \\ 1 & -2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{bmatrix} = \begin{bmatrix} -2/3-16/3+7 & -4/3-20/3+8 & -2-8+10 \\ -2/3+44/3-14 & -4/3+55/3-16 & -2+22-20 \\ 1-8+7 & 2-10+8 & 3-12+10 \end{bmatrix}$$

## Or

$$\mathbf{A}^{-1}\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# MATLAB and True BASIC Do Not Require Inverse Formulas

## MATLAB

```
A=[1 2 3;4 5 6;7 8 10];
AINV=inv(A)
```

## True BASIC

```
OPTION NOLET
DIM A(3,3),AINV(3,3)
A(1,1)=1.
A(1,2)=2.
A(1,3)=3.
A(2,1)=4.
A(2,2)=5.
A(2,3)=6.
A(3,1)=7.
A(3,2)=8.
A(3,3)=10.
MAT AINV=INV(A)
MAT PRINT AINV
END
```

# Numerical Integration of Differential Equations

# Euler Integration

## Given first-order differential equation

$$\dot{\mathbf{x}} = f(\mathbf{x},t)$$

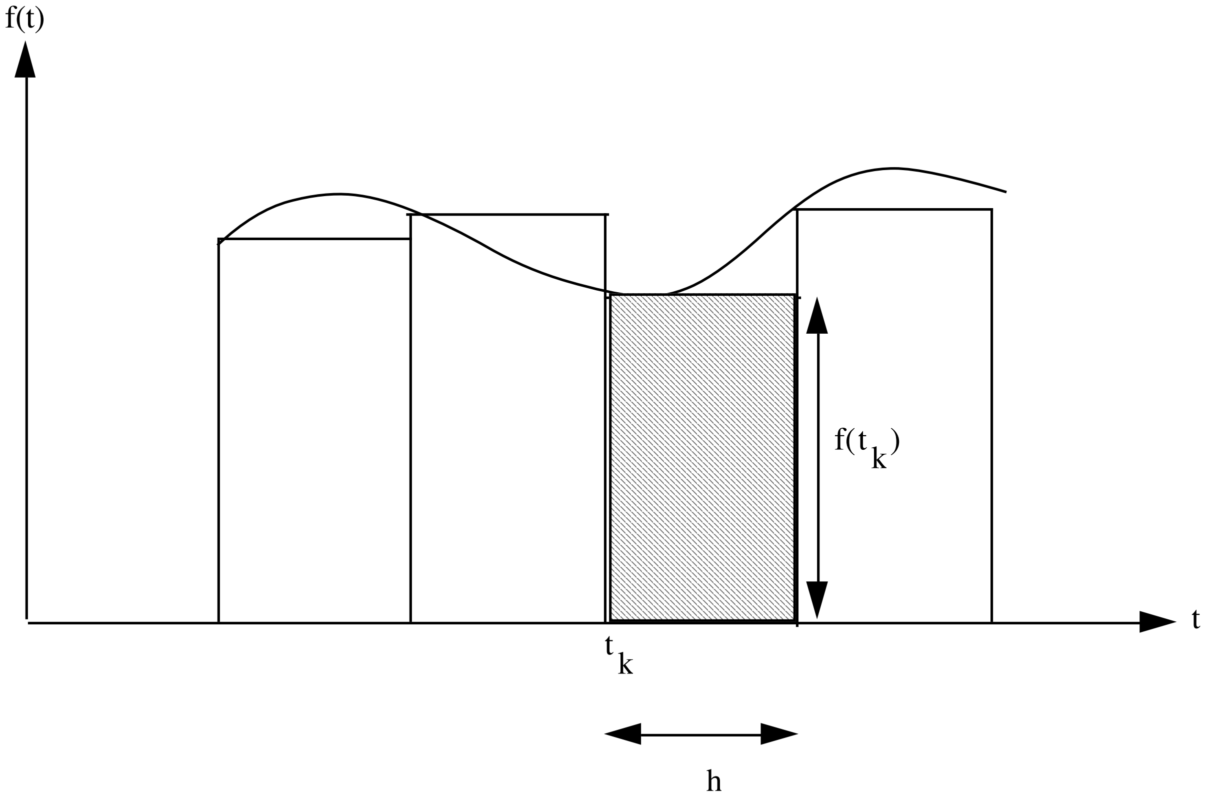
## From the definition of a derivative in calculus

$$\dot{\mathbf{x}} = f(\mathbf{x},t) = \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} = \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{h}$$

## Rearranging terms

$$\mathbf{x}_k = \mathbf{x}_{k-1} + hf(\mathbf{x},t) \longleftarrow \text{Euler integration}$$

# Finding Area Under Curve is Equivalent to Euler Integration



# Making Up a Differential Equation To Test Euler Integration

**Answer**

$$x = \sin\omega t$$

**Take first derivative**

$$\dot{x} = \omega\cos\omega t$$

**Take second derivative**

$$\ddot{x} = -\omega^2\sin\omega t$$

**Therefore**

$$\ddot{x} = -\omega^2 x \longleftarrow$$ **Second-order differential equation we want to solve**

**Initial conditions**

$$\left.\begin{aligned} x(0) &= 0 \\ \dot{x}(0) &= \omega \end{aligned}\right]$$ **Obtained from first and second equations at t=0**

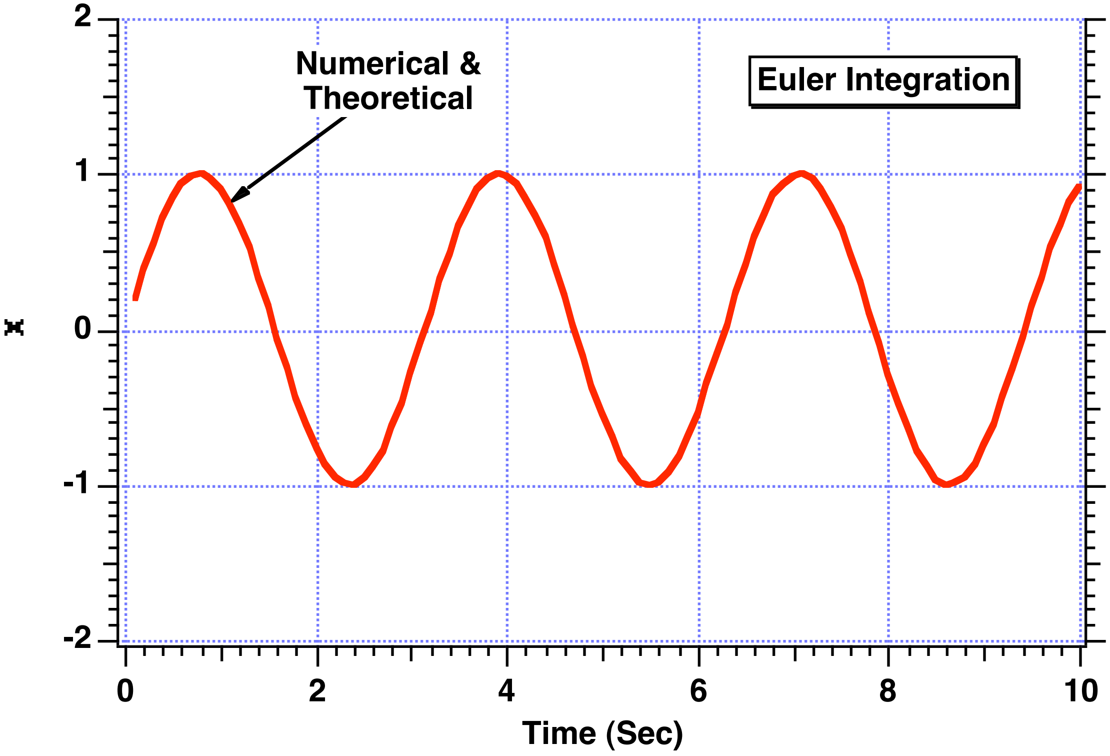
# Using Euler Integration to Solve Second-Order Differential Equation

```
IMPLICIT REAL*8(A-H)
IMPLICIT REAL*8(O-Z)
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
W=2.
T=0.
S=0.
X=0.
XD=W
H=.01
WHILE(T<=10.)
        S=S+H
        XDD=-W*W*X
        XD=XD+H*XDD
        X=X+H*XD
        T=T+H
        IF(S>=.09999)THEN
                S=0.
                XTHEORY=SIN(W*T)
                WRITE(9,*)T,X,XTHEORY
                WRITE(1,*)T,X,XTHEORY
        ENDIF
END DO
PAUSE
CLOSE(1)
END
```

**Initial conditions**

**Differential equation and Euler integration**

# Euler Integration Accurately Solves Second-Order Differential Equation



# Second-Order Runge-Kutta Integration

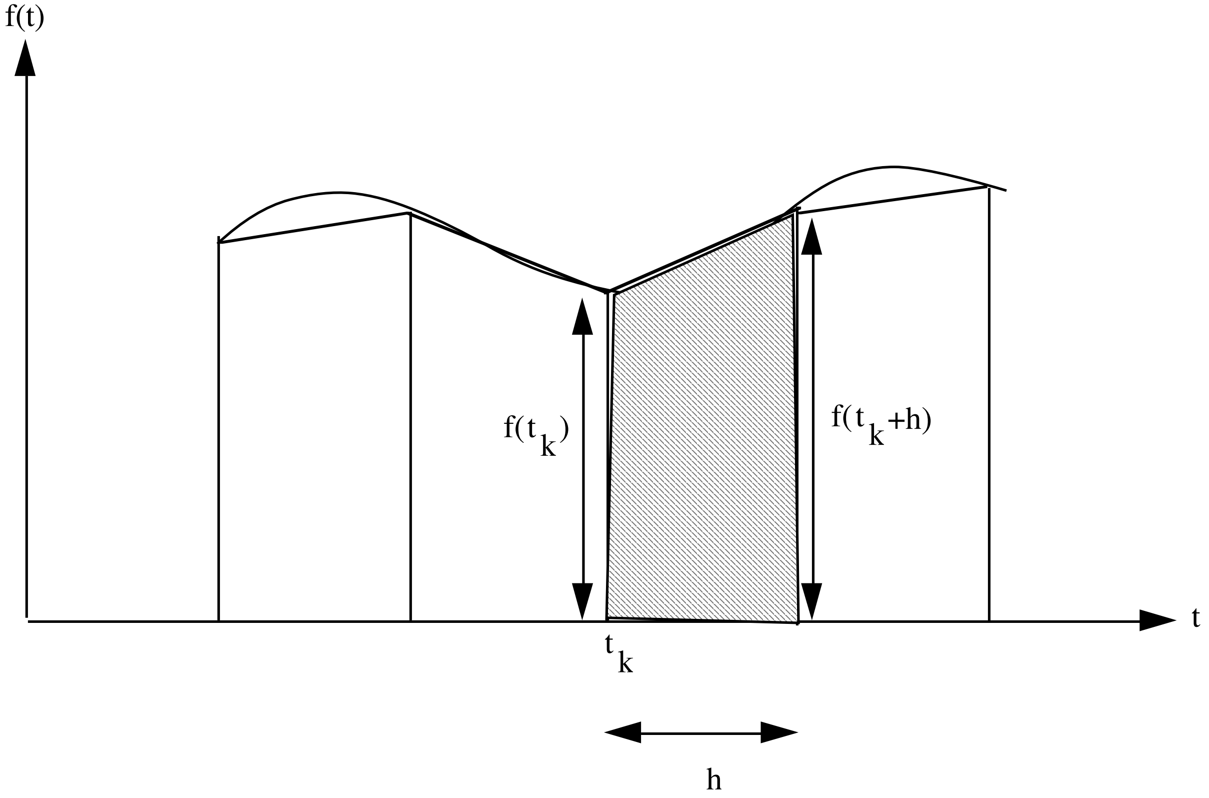
## Given

$$\dot{\mathbf{x}} = f(\mathbf{x},t)$$

## Second-order Runge-Kutta formula

$$\mathbf{x}_k = \mathbf{x}_{k-1} + .5h[f(\mathbf{x},t) + f(\mathbf{x},t+h)]$$

## Equivalent to using trapezoids to find area under curve



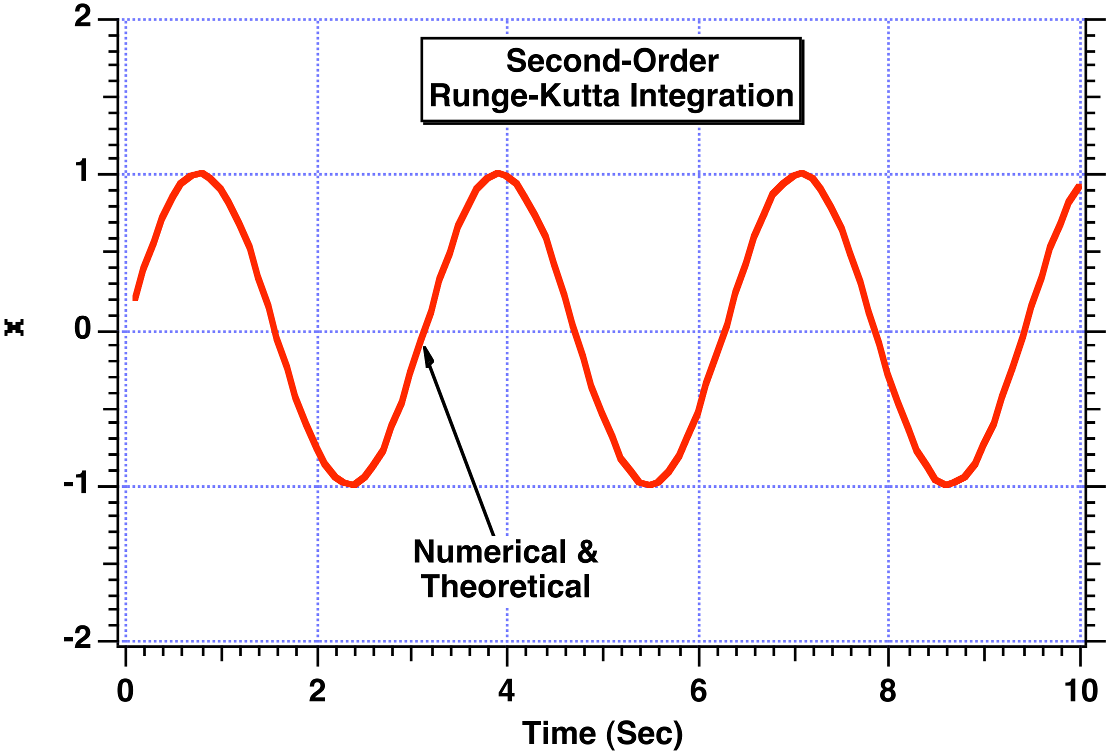
# Using Second-Order Runge-Kutta Integration to Solve Same Differential Equation

```
IMPLICIT REAL*8(A-H)
IMPLICIT REAL*8(O-Z)
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
W=2.
T=0.
S=0.
X=0.
XD=W
H=.01
WHILE(T<=10.)
        S=S+H
        XOLD=X
        XDOLD=XD
        XDD=-W*W*X
        X=X+H*XD
        XD=XD+H*XDD
        T=T+H
        XDD=-W*W*X
        X=.5*(XOLD+X+H*XD)
        XD=.5*(XDOLD+XD+H*XDD)
        IF(S>=.09999)THEN
                S=0.
                XTHEORY=SIN(W*T)
                WRITE(9,*)T,X,XTHEORY
                WRITE(1,*)T,X,XTHEORY
        ENDIF
END DO
PAUSE
CLOSE(1)
END
```

**Initial conditions**

**Second-order Runge-Kutta integration**

# Second-Order Runge-Kutta Numerical Integration Also Accurately Solves Differential Equation



# Noise and Random Variables

# Basic Definitions - 1

## Probability density function

$$p(x) \geq 0$$

$$\int_{-\infty}^{\infty} p(x)dx = 1$$

## Probability that x is between a and b

$$\mathrm{Prob}(a \leq x \leq b) = \int_{a}^{b} p(x)dx$$

## Distribution function

$$P(x) = \int_{-\infty}^{x} p(u)du$$

# Basic Definitions - 2

## Mean or expected value

$$m = E(x) = \int_{-\infty}^{\infty} xp(x)dx$$

## Property

$$E(x_1 + x_2 + ... + x_n) = E(x_1) + E(x_2) + ... + E(x_n)$$

## Mean squared value

$$E(x^2) = \int_{-\infty}^{\infty} x^2p(x)dx$$

## Root mean square value

$$rms = \sqrt{E(x^2)}$$

# Basic Definitions - 3

## Variance

$$\sigma^2 = E\{[x - E(x)]^2\} = E(x^2) - E^2(x)$$

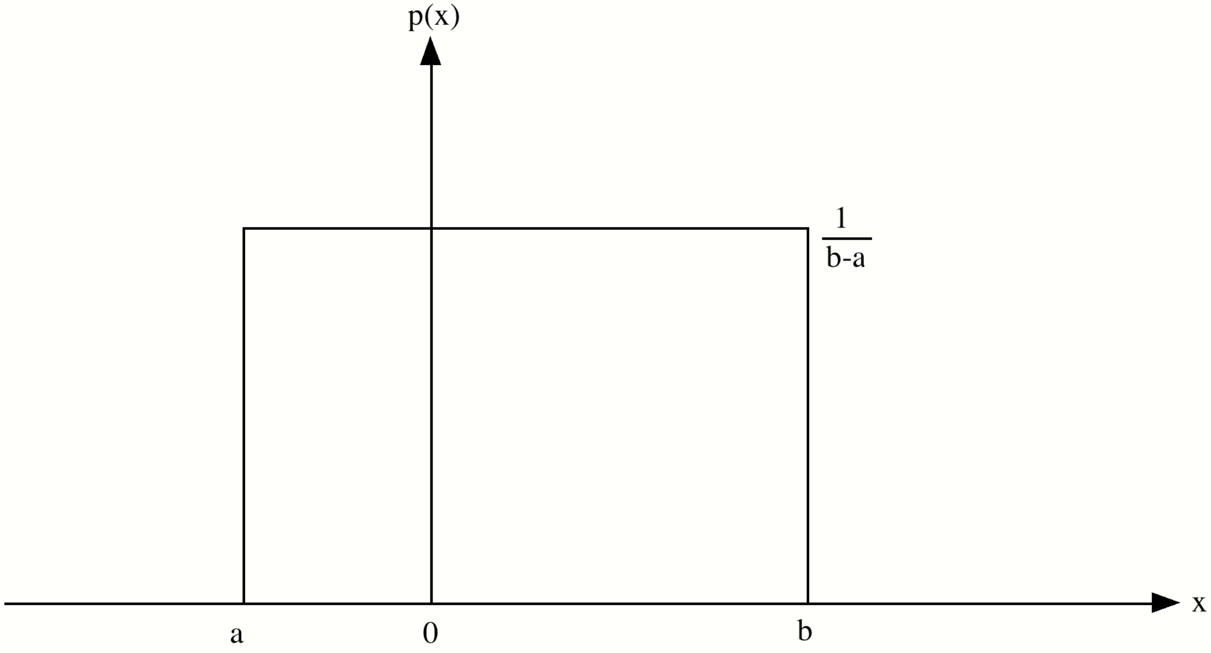
## Property for independent random variables

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 + ... + \sigma_n^2$$

**Square root of variance is known as standard deviation**

**For zero mean processes the standard deviation and RMS values are identical**

# Uniform Probability Distribution



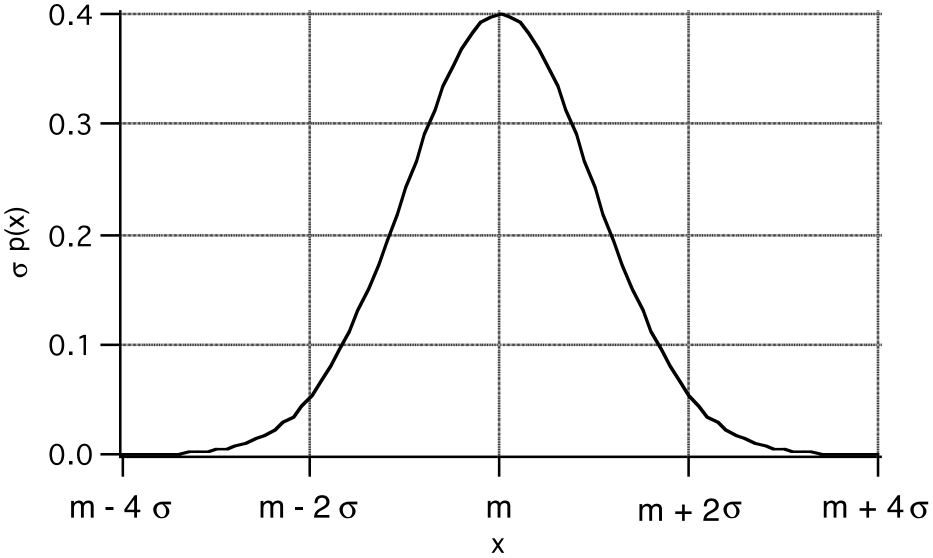
## Mean

$$m = E(x) = \int_{-\infty}^{\infty} xp(x)dx = \frac{1}{b-a}\int_{a}^{b} xdx = \frac{b+a}{2}$$

## Variance

$$\sigma^2 = E(x^2) - E^2(x) = \frac{b^3 - a^3}{3(b - a)} - \left(\frac{b + a}{2}\right)^2 = \frac{(b - a)^2}{12}$$

# Gaussian or Normal Probability Density Function



## Probability density function

$$p(x) = \frac{\exp\left[-\frac{(x-m)^2}{2\sigma^2}\right]}{\sigma\sqrt{2\pi}}$$

# Gaussian Random Noise Generator in FORTRAN

```
C THE FIRST THREE STATEMENTS INVOKE THE ABSOFT RANDOM NUMBER GENERATOR ON THE MACINTOSH
         GLOBAL DEFINE
                 INCLUDE 'quickdraw.inc'
         END
         IMPLICIT REAL*8 (A-H)
         IMPLICIT REAL*8 (O-Z)
         SIGNOISE=1.
         OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
         DO 10 I=1,1000
         CALL GAUSS(X,SIGNOISE)
         WRITE(9,*)I,X
         WRITE(1,*)I,X
 10      CONTINUE
         CLOSE(1)
         PAUSE
         END

         SUBROUTINE GAUSS(X,SIG)
         IMPLICIT REAL*8(A-H)
         IMPLICIT REAL*8(O-Z)
         INTEGER SUM
         SUM=0
         DO 14 J=1,6
C THE NEXT STATEMENT PRODUCES A UNIF. DISTRIBUTED NUMBER FROM -32768 TO +32768
         IRAN=Random()
         SUM=SUM+IRAN
 14      CONTINUE
         X=SUM/65536.
         X=1.414*X*SIG
         RETURN
         END
```

# Gaussian Random Number Generator in MATLAB and True BASIC

## MATLAB

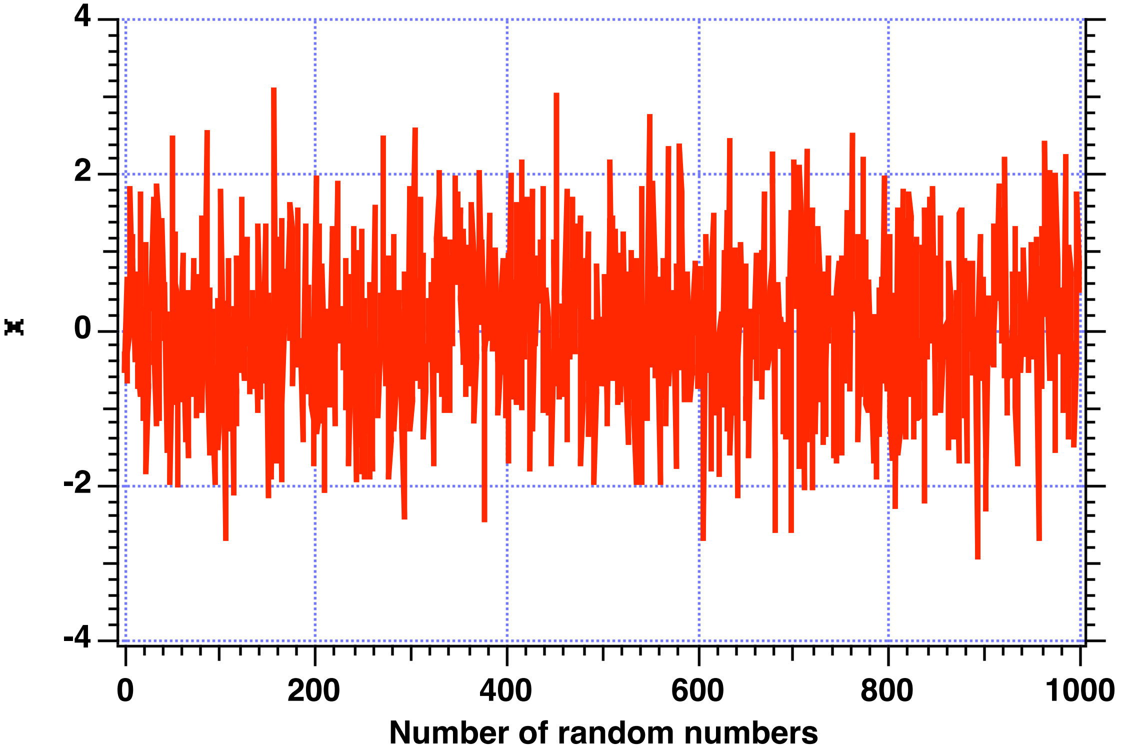
```
SIGNOISE=1;
count=0;
for I=1:1000;
          X=SIGNOISE*randn;
          count=count+1;
          ArrayI(count)=I;
          ArrayX(count)=X;
end
clc
output=[ArrayI',ArrayX'];
save datfil output  -ascii
disp 'simulation finished'
```

## True BASIC

```
OPTION NOLET
REM UNSAVE "DATFIL"
OPEN #1:NAME "DATFIL",ACCESS OUTPUT,CREATE NEW, ORGANIZATION TEXT
SET #1: MARGIN 1000
SIGNOISE=1.
FOR I=1 TO 1000
          CALL GAUSS(X,SIGNOISE)
          PRINT I,X
          PRINT #1:I,X
NEXT I
CLOSE #1
END

SUB GAUSS(X,SIG)
LET X=RND+RND+RND+RND+RND+RND-3
LET X=1.414*X*SIG
END SUB
```

# One Thousand Random Numbers With Gaussian Distribution



**Standard deviation appears to be correct**

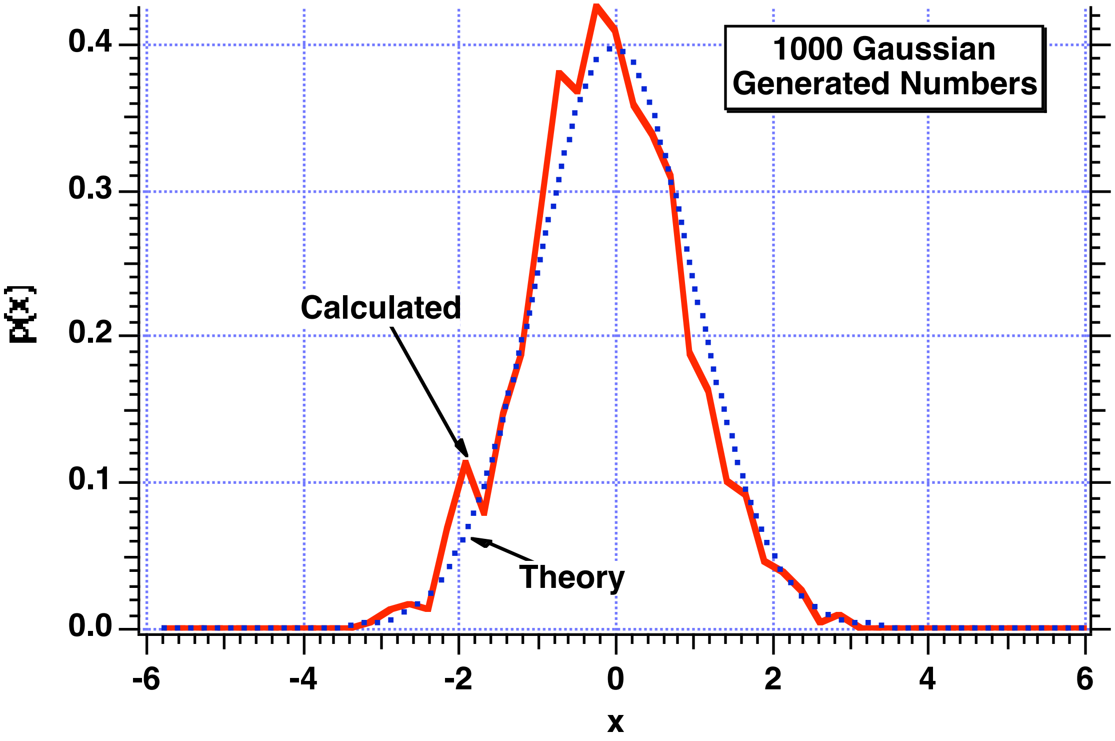
$$\sigma_{\text{APPROX}} \approx \frac{\text{Peak to Peak}}{6} \approx \frac{6}{6} = 1$$

# Program to Generate Probability Density Function

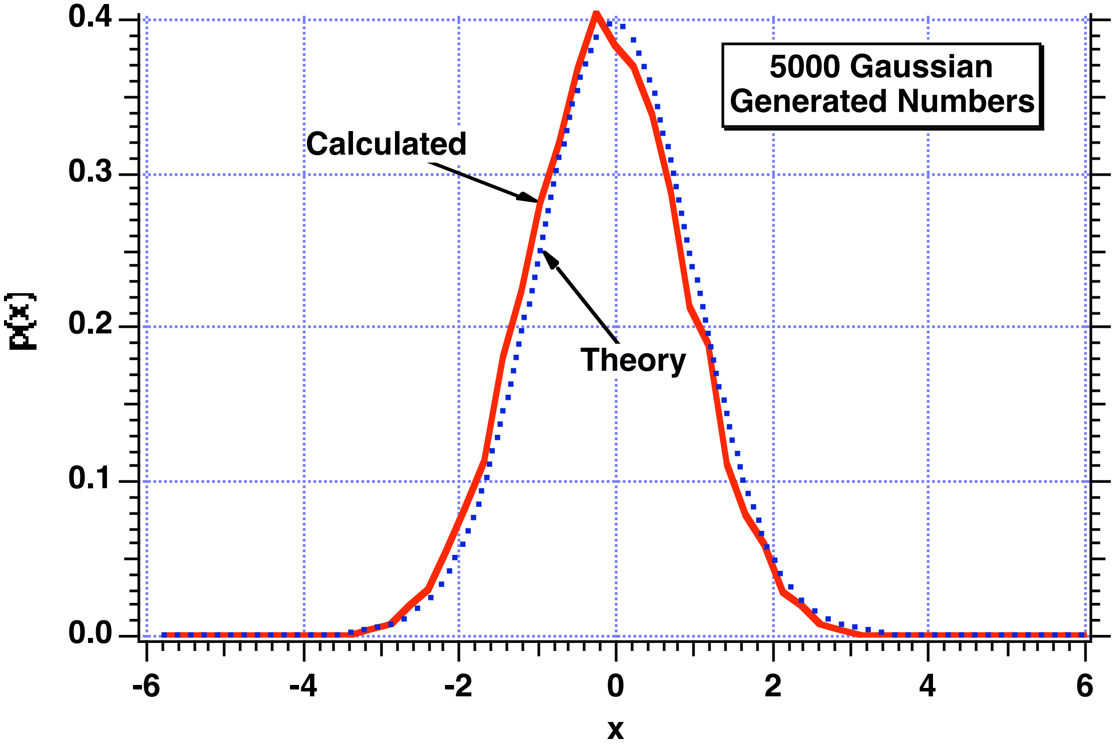
```
         GLOBAL DEFINE
                   INCLUDE 'quickdraw.inc'
         END
         IMPLICIT REAL*8 (A-H)
         IMPLICIT REAL*8 (O-Z)
         INTEGER BIN
         REAL*8 H(2000),X(2000)
         OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
         XMAX=6.
         XMIN=-6.
         SIGNOISE=1.
         RANGE=XMAX-XMIN
         TMP=1./SQRT(6.28)
         BIN=50
         N=1000
         DO 10 I=1,N
         CALL GAUSS(Y,SIGNOISE)
         X(I)=Y
10       CONTINUE
         DO 20 I=1,BIN
         H(I)=0
20       CONTINUE
         DO 30 I=1,N
         K=INT(((X(I)-XMIN)/RANGE)*BIN)+.99
         IF(K<1)K=1
         IF(K>BIN)K=BIN
         H(K)=H(K)+1
30       CONTINUE
         DO 40 K=1,BIN
         PDF=(H(K)/N)*BIN/RANGE
         AB=XMIN+K*RANGE/BIN
         TH=TMP*EXP(-AB*AB/2.)
         WRITE(9,*)AB,PDF,TH
         WRITE(1,*)AB,PDF,TH
40       CONTINUE
         PAUSE
         CLOSE(1)
         END
```

**Generate 1000 random numbers with Gaussian distribution**

# Sample Gaussian Distribution Matches Theoretical Distribution For 1000 Random Numbers



# Sample Gaussian Distribution Matches Theoretical Distribution Even Better For 5000 Random Numbers



# Calculating Random Variable Properties From a Finite Set of Data

$$\text{mean} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\text{mean square} = \frac{\sum_{i=1}^{n} x_i^2}{n - 1}$$

$$\text{standard deviation} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \text{mean})^2}{n - 1}}$$

# White Noise

## Autocorrelation function

$$\phi_{xx}(t_1,t_2) = E[x(t_1)x(t_2)]$$

## Power spectral density in units squared per Hertz

$$\Phi_{xx} = \int_{-\infty}^{\infty} \phi_{xx}(\tau)e^{-j\omega\tau}d\tau$$

## Power spectral density of white noise is constant

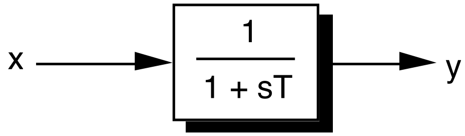
$$\Phi_{xx} = \Phi_0 \qquad \text{(white noise)}$$

## Autocorrelation function of white noise is an impulse

$$\phi_{xx} = \Phi_0\delta(\tau) \qquad \text{(white noise)}$$

# Example of Simulating White Noise

## Low-pass filter driven by white noise



## Resultant differential equation

$$\frac{y}{x} = \frac{1}{1 + sT} \longrightarrow y + \dot{y}T = x \longrightarrow \dot{y} = \frac{(x - y)}{T}$$

## One can show that

$$E[y^2(t)] = \frac{\Phi_0(1 - e^{-2t/T})}{2T} \longleftarrow$$ **Theoretical answer**

**We can simulate pseudo white noise by adding Gaussian noise every integration interval with**

$$\sigma = \sqrt{\frac{\Phi_0}{h}} \longleftarrow$$ **Very important**

# Simulation of Low-Pass Filter Driven By White Noise

```
GLOBAL DEFINE
          INCLUDE 'quickdraw.inc'
END
IMPLICIT REAL*8 (A-H)
IMPLICIT REAL*8 (O-Z)
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
TAU=.2
PHI=1.
T=0.
H=.01
SIG=SQRT(PHI/H)
Y=0.
WHILE(T<=4.999)
          CALL GAUSS(X,SIG)
          YOLD=Y
          YD=(X-Y)/TAU
          Y=Y+H*YD
          T=T+H
          YD=(X-Y)/TAU
          Y=(YOLD+Y)/2.+.5*H*YD
          SIGPLUS=SQRT(PHI*(1.-EXP(-2.*T/TAU))/(2.*TAU))
          SIGMINUS=-SIGPLUS
          WRITE(9,*)T,Y,SIGPLUS,SIGMINUS
          WRITE(1,*)T,Y,SIGPLUS,SIGMINUS
END DO
PAUSE
CLOSE(1)
END
```

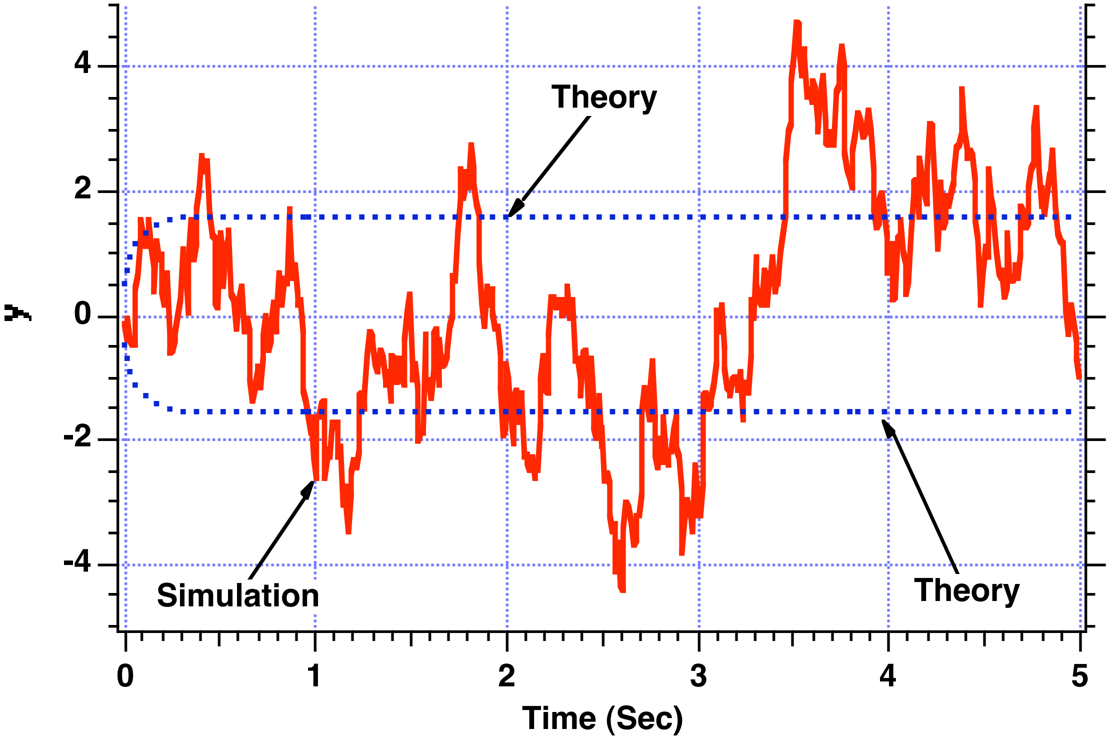
**Desired spectral density of white noise**

**Standard deviation of pseudo white noise**

**Pseudo white noise**

**Second-order Runge-Kutta integration**

# Low-Pass Filter Output Agrees With Theory



# State Space Notation

# First-Order Example of State Space Notation

## General form

$$\dot{\mathbf{x}} = \mathbf{Fx} + \mathbf{Gu} + \mathbf{w}$$

## Low-pass filter example

$$\dot{y} = \frac{(x - y)}{T}$$

## Change notation to avoid confusion

$$\dot{x} = \frac{(n - x)}{T}$$

## State space matrices are all scalars

$$\mathbf{F} = \frac{-1}{T}$$

$$\mathbf{G} = 0$$

$$\mathbf{w} = \frac{n}{T}$$

# Second-Order Example of State Space Notation

## General form

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} + \mathbf{w}$$

## Second-order differential equation

$$\ddot{y} + 2\dot{y} + 3y = 4$$

## Solve for highest derivative

$$\ddot{y} = -2\dot{y} - 3y + 4$$

## Express in matrix form

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} 4$$

## By comparison state space matrices are

$$\mathbf{x} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \qquad \mathbf{u} = 4$$

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

# Fundamental Matrix

# Definition of Fundamental Matrix

Given a system described by

$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x}$ ← F is time invariant

There exists fundamental matrix to propagate states forward

$$\mathbf{x}(t) = \mathbf{\Phi}(t - t_0)\mathbf{x}(t_0)$$

Two ways of finding fundamental matrix

$\mathbf{\Phi}(\mathbf{t}) = £^{-1}[(s\mathbf{I} - \mathbf{F})^{-1}]$ ← Laplace transform method

$\mathbf{\Phi}(t) = e^{\mathbf{F}t} = \mathbf{I} + \mathbf{F}t + \frac{(\mathbf{F}t)^2}{2!} + ... + \frac{(\mathbf{F}t)^n}{n!} + ...$ ← Taylor series expansion

# Example of Laplace Transform Method For Finding Fundamental Matrix - 1

We have already shown that solution to

$\ddot{x} = -\omega^2 x$ With initial conditions $x(0) = 0$ $\dot{x}(0) = \omega$

Is given by

$$x = \sin\omega t$$

And taking derivative we get

$$\dot{x} = \omega\cos\omega t$$

Rewriting original differential equation in state space form yields

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \longrightarrow \dot{\mathbf{x}} = \mathbf{Fx} + \mathbf{Gu} + \mathbf{w}$$

Systems dynamics matrix given by

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix}$$

# Example of Laplace Transform Method For Finding Fundamental Matrix - 2

## Recall

$$\boldsymbol{\Phi}(\mathbf{t}) = £^{-1}[(\mathbf{sI} - \mathbf{F})^{-1}]$$

## Substitution yields

$$(\mathbf{sI} - \mathbf{F})^{-1} = \begin{bmatrix} s & -1 \\ \omega^2 & s \end{bmatrix}^{-1}$$

## Using formulas for two by two inverse yields

$$\boldsymbol{\Phi}(\mathbf{s}) = (\mathbf{sI} - \mathbf{F})^{-1} = \frac{1}{s^2+\omega^2}\begin{bmatrix} s & 1 \\ -\omega^2 & s \end{bmatrix}$$

## From inverse Laplace transform tables

$$\boldsymbol{\Phi}(\mathbf{t}) = \begin{bmatrix} \cos\omega t & \frac{\sin\omega t}{\omega} \\ -\omega\sin\omega t & \cos\omega t \end{bmatrix}$$

# Checking Fundamental Matrix Solution

## Initial conditions to differential equation

$$x(0) = \sin\omega(0) = 0$$

$$\dot{x}(0) = \omega\cos\omega(0) = \omega$$

## Since

$$\mathbf{x}(t) = \mathbf{\Phi}(t - t_0)\mathbf{x}(t_0)$$

## We can also say that

$$\mathbf{x}(t) = \mathbf{\Phi}(t)\mathbf{x}(0)$$

## Substitution yields

$$\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} \cos\omega t & \dfrac{\sin\omega t}{\omega} \\ -\omega\sin\omega t & \cos\omega t \end{bmatrix} \begin{bmatrix} x(0) \\ \dot{x}(0) \end{bmatrix} = \begin{bmatrix} \cos\omega t & \dfrac{\sin\omega t}{\omega} \\ -\omega\sin\omega t & \cos\omega t \end{bmatrix} \begin{bmatrix} 0 \\ \omega \end{bmatrix} = \begin{bmatrix} \sin\omega t \\ \omega\cos\omega t \end{bmatrix}$$

## Or

$$x(t) = \sin\omega t$$

$$\dot{x}(t) = \omega\cos\omega t$$

## Which are the correct solutions obtained without integration!

# Using Taylor Series Method For Finding Fundamental Matrix - 1

## Recall

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix}$$

## Therefore

$$\mathbf{F}^2 = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} = \begin{bmatrix} -\omega^2 & 0 \\ 0 & -\omega^2 \end{bmatrix}$$

$$\mathbf{F}^3 = \begin{bmatrix} -\omega^2 & 0 \\ 0 & -\omega^2 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\omega^2 \\ \omega^4 & 0 \end{bmatrix}$$

$$\mathbf{F}^4 = \begin{bmatrix} 0 & -\omega^2 \\ \omega^4 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} = \begin{bmatrix} \omega^4 & 0 \\ 0 & \omega^4 \end{bmatrix}$$

$$\mathbf{F}^5 = \begin{bmatrix} \omega^4 & 0 \\ 0 & \omega^4 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \omega^4 \\ -\omega^6 & 0 \end{bmatrix}$$

$$\mathbf{F}^6 = \begin{bmatrix} 0 & \omega^4 \\ -\omega^6 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} = \begin{bmatrix} -\omega^6 & 0 \\ 0 & -\omega^6 \end{bmatrix}$$

# Using Taylor Series Method For Finding Fundamental Matrix - 2

## Truncating Taylor series to 6 terms yields

$$\boldsymbol{\Phi}(t) = e^{\mathbf{F}t} \approx \mathbf{I} + \mathbf{F}t + \frac{(\mathbf{F}t)^2}{2!} + \frac{(\mathbf{F}t)^3}{3!} + \frac{(\mathbf{F}t)^4}{4!} + \frac{(\mathbf{F}t)^5}{5!} + \frac{(\mathbf{F}t)^6}{6!}$$

## Or

$$\boldsymbol{\Phi}(t) \approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} t + \begin{bmatrix} -\omega^2 & 0 \\ 0 & -\omega^2 \end{bmatrix} \frac{t^2}{2} + \begin{bmatrix} 0 & -\omega^2 \\ \omega^4 & 0 \end{bmatrix} \frac{t^3}{6} +$$
$$\begin{bmatrix} \omega^4 & 0 \\ 0 & \omega^4 \end{bmatrix} \frac{t^4}{24} + \begin{bmatrix} 0 & \omega^4 \\ -\omega^6 & 0 \end{bmatrix} \frac{t^5}{120} + \begin{bmatrix} -\omega^6 & 0 \\ 0 & -\omega^6 \end{bmatrix} \frac{t^6}{720}$$

## Combining terms

$$\boldsymbol{\Phi}(t) \approx \begin{bmatrix} 1 - \frac{\omega^2 t^2}{2} + \frac{\omega^4 t^4}{24} - \frac{\omega^6 t^6}{720} & t - \frac{\omega^2 t^3}{6} + \frac{\omega^4 t^5}{120} \\ -\ \omega^2 t + \frac{\omega^4 t^3}{6} - \frac{\omega^6 t^5}{120} & 1 - \frac{\omega^2 t^2}{2} + \frac{\omega^4 t^4}{24} - \frac{\omega^6 t^6}{720} \end{bmatrix}$$

## Recognizing that

$$\sin\omega t \approx \omega t - \frac{\omega^3 t^3}{3!} + \frac{\omega^5 t^5}{5!} - \ldots$$

$$\cos\omega t \approx 1 - \frac{\omega^2 t^2}{2!} + \frac{\omega^4 t^4}{4!} - \frac{\omega^6 t^6}{6!} + \ldots$$

# Using Taylor Series Method For Finding Fundamental Matrix - 3

We get

$$\boldsymbol{\Phi}(\mathbf{t}) = \begin{bmatrix} \cos\omega t & \frac{\sin\omega t}{\omega} \\ -\omega\sin\omega t & \cos\omega t \end{bmatrix}$$

Which is the same answer obtained with the Laplace transform method

# Check of Fundamental Matrix

If our model of the real world is given by

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{g}\mathbf{u} + \mathbf{w}$$

The continuous fundamental matrix can also be found by solving

$$\dot{\Phi} = \mathbf{F}\Phi, \quad \Phi(0) = \mathbf{I}$$

As an example we already know that for

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

The continuous fundamental matrix is given by

$$\Phi(t) = \begin{bmatrix} \cos\omega t & \dfrac{\sin\omega t}{\omega} \\ -\omega \sin\omega t & \cos\omega t \end{bmatrix}$$

# Integrating Matrix Differential Equation-1

```
        IMPLICIT REAL*8(A-H,O-Z)

        REAL*8 F(2,2),PHI(2,2),PHIOLD(2,2),PHID(2,2)
        INTEGER ORDER
        OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
        ORDER=2
        W=3.
        TF=10.
        T=0.
        S=0.
        DO 14 I=1,ORDER
        DO 14 J=1,ORDER
        F(I,J)=0.
        PHI(I,J)=0.
14      CONTINUE
        F(1,2)=1.
        F(2,1)=-W*W
        PHI(1,1)=1.
        PHI(2,2)=1.
        H=.01
        WHILE(T<=TF)
                DO 20 I=1,ORDER
                DO 20 J=1,ORDER
                        PHIOLD(I,J)=PHI(I,J)
20              CONTINUE
```

**F and $\Phi_0$**

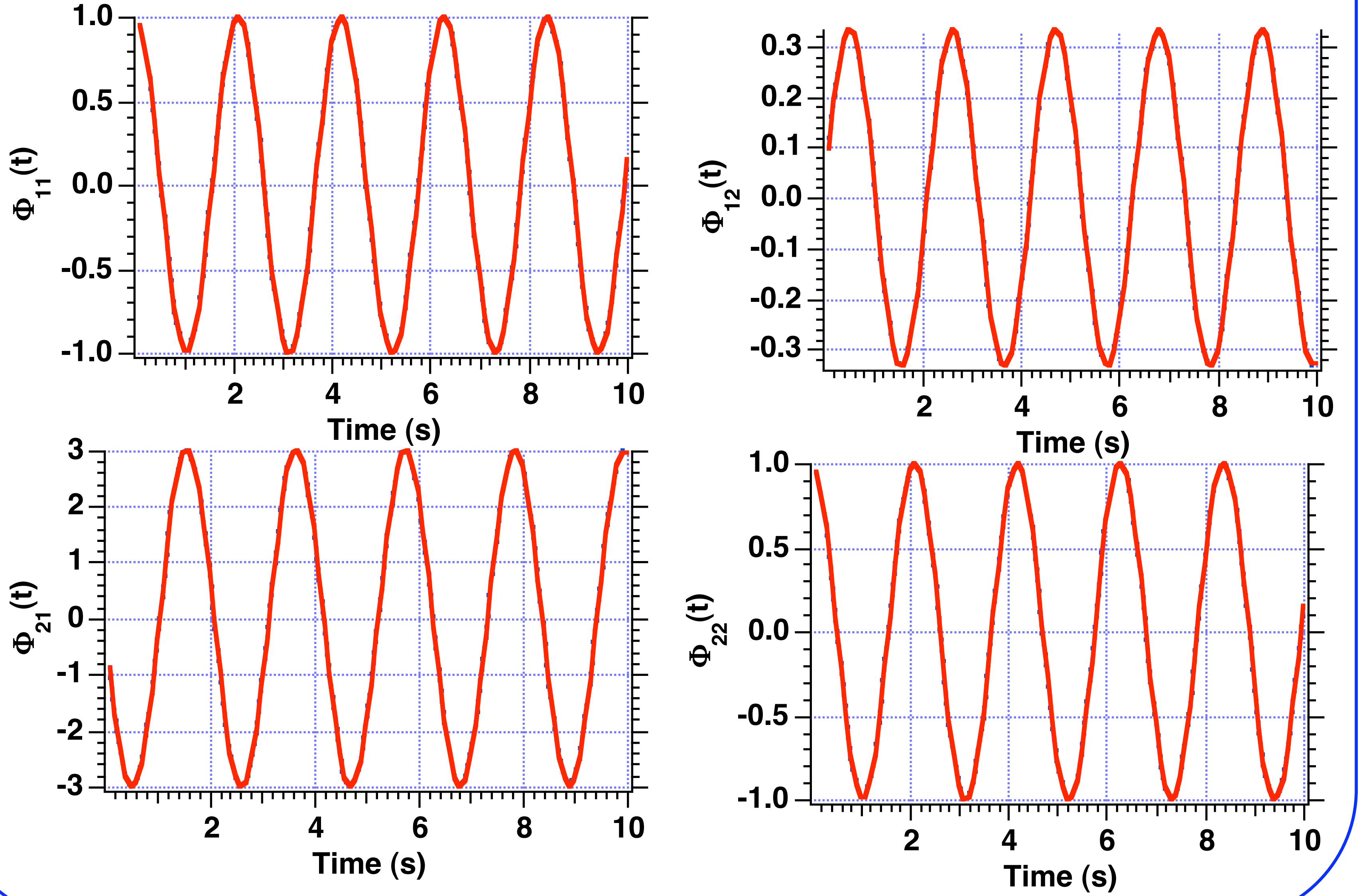
**Second-Order Runge-Kutta Integration**

# Integrating Matrix Differential Equation-2

```
          CALL MATMUL(F,ORDER,ORDER,PHI,ORDER,ORDER,PHID)
                    DO 50 I=1,ORDER
                    DO 50 J=1,ORDER
                              PHI(I,J)=PHI(I,J)+H*PHID(I,J)
50                  CONTINUE
                    T=T+H
                    CALL MATMUL(F,ORDER,ORDER,PHI,ORDER,ORDER,PHID)
                    DO 60 I=1,ORDER
                    DO 60 J=1,ORDER
                              PHI(I,J)=.5*(PHIOLD(I,J)+PHI(I,J)+H*PHID(I,J))
60                  CONTINUE
                    S=S+H
                    IF(S>=.09999)THEN
                              S=0.
                              P11TH=COS(W*T)
                              P12TH=SIN(W*T)/W
                              P21TH=-W*SIN(W*T)
                              P22TH=COS(W*T)
                              WRITE(9,*)T,PHI(1,1),P11TH,PHI(1,2),P12TH,
  1                                     PHI(2,1),P21TH,PHI(2,2),P22TH
                              WRITE(1,*)T,PHI(1,1),P11TH,PHI(1,2),P12TH,
  1                                     PHI(2,1),P21TH,PHI(2,2),P22TH
                    ENDIF
          END DO
          PAUSE
          CLOSE(1)
          END
```

**Output Matrix Elements**

# Simulated and Calculated Elements of Fundamental Matrix Agree



# Numerical Basics
# Summary

- Vector and matrix manipulations introduced and demonstrated
- Numerical integration techniques presented and verified
- Source code can easily be converted to other languages
- State space concepts and fundamental matrix introduced