Verification and Validation

Tutorial 02

Prof. Americo Cunha Jr

Rio de Janeiro State University – UERJ

americo.cunha@uerj.br

www.americocunha.org

@AmericoCunhaJr @AmericoCunhaJr

@AmericoCunhaJr @AmericoCunhaJr

Verification and Validation (V&V)

- Verification
  Are we solving the equation *right*?

- Validation
  Are we solving the *right* equation?

G. Iaccarino *Quantification of Uncertainty in Flow Simulations Using Probabilistic Methods*,
VKI Lecture Series, Stanford University, 2008

Verification and Validation (V&V)

- Verification
  Are we solving the equation *right*?
  It is an exercise in *mathematics*.

- Validation
  Are we solving the *right* equation?

G. Iaccarino *Quantification of Uncertainty in Flow Simulations Using Probabilistic Methods*,
VKI Lecture Series, Stanford University, 2008

Verification and Validation (V&V)

- Verification
  Are we solving the equation *right*?
  It is an exercise in *mathematics*.

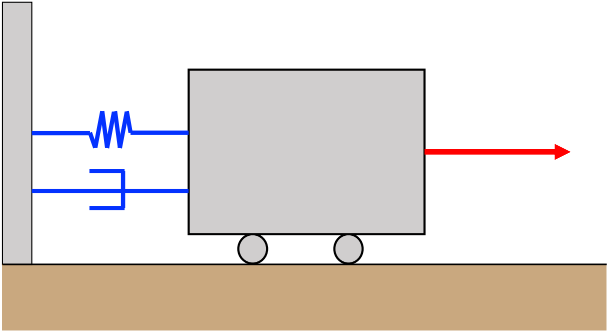
- Validation
  Are we solving the *right* equation?
  It is an exercise in *physics*.

G. Iaccarino *Quantification of Uncertainty in Flow Simulations Using Probabilistic Methods*,

VKI Lecture Series, Stanford University, 2008

V&V for Mass-Spring-Damper Oscillator

Mass-Spring-Damper Oscillator



$$\ddot{x} + 2\,\xi\,\omega_n\,\dot{x} + \omega_n^2\,x = (f/m)\sin(\omega\,t)$$

$$\dot{x}(0) = v_0, \quad x(0) = x_0$$

Model equation

$$\ddot{x} + 2\,\xi\,\omega_n\,\dot{x} + \omega_n^2\,x = (f/m)\sin(\omega\,t)$$

$$\Longleftrightarrow$$

$$\underbrace{\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix}}_{\dot{\phi}} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\,\xi\,\omega_n \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} 0 \\ (f/m)\,\sin(\omega\,t) \end{bmatrix}}_{f(t,\phi)}$$

where $\phi_1 = x$ and $\phi_2 = \dot{x}$.

Reference solution

The unforced case, that corresponds to $f = 0$, has an analytical solution, which is given by

$$x = A\, e^{-\xi \omega_n t}\, \sin\left(\omega_d\, t + \phi\right),$$

where

$$\omega_d = \omega_n \sqrt{1 - \xi^2},$$

$$A = \sqrt{x_0^2 + \left(\frac{v_0 + \xi\, \omega_n\, x_0}{\omega_d}\right)^2},$$

$$\phi = \tan^{-1}\left(\frac{x_0\, \omega_d}{v_0 + \xi\, \omega_n\, x_0}\right).$$

This solution will be used as reference in <u>Verification</u> step.

Numerical method

The initial value problem can be written as

$$\dot{\phi} = f\left(t, \phi\right), \quad \phi(0) = \phi_0$$

where $\phi = \left[\begin{array}{cc} x & \dot{x} \end{array} \right]^T$.

Numerical integration will be done via Explicit Euler method

$$\phi_{n+1} = \phi_n + \Delta t \, f(t_n, \phi_n)$$

$$t_{n+1} = t_n + \Delta t$$

where ϕ_n is an approximation for $\phi(t_n)$.

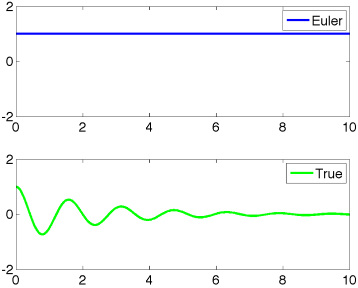
main_verification11.m

```
clc; clear all; close all;
m = 1.0; ksi = 0.1; wn = 4.0; f = 0; w = 5.0;
x0 = 1.0; v0 = 0.0; t0 = 0.0; t1 = 10.0; N = 3000;

dphidt=@(t,phi)[0 1; -wn^2 -2*ksi*wn]*phi ...
             + [0; (f/m)*sin(w*t)];

[time,phi] = euler(dphidt,[x0;v0],t0,t1,N);

wd     = wn*sqrt(1-ksi^2);
A      = sqrt(x0^2 + ((v0+ksi*wn*x0)/wd)^2);
theta  = atan((x0*wd)/(v0+ksi*wn*x0));
x_true = A*exp(-ksi*wn*time).*sin(wd*time+theta);

subplot(2,1,1)
plot(time,phi(1,:),'b','LineWidth',3);
legend('Euler')
subplot(2,1,2)
plot(time,x_true,  'g','LineWidth',3);
legend('True')
```

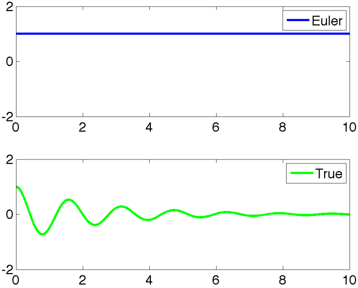
euler.m

```matlab
function [time,phi] = euler(rhs,phi0,t0,t1,N)

dt        = (t1-t0)/N;
time      = zeros(1,N+1);
phi       = zeros(length(phi0),N+1);
phi(:,1)  = phi0;

for n = 1:N
  time(1,n+1) = time(1,n) + dt;
  phi(:,n+1) =  phi(:,n) + dt*rhs(t0,phi0);
end

return
```

Verification of the equation solution (unforced case)

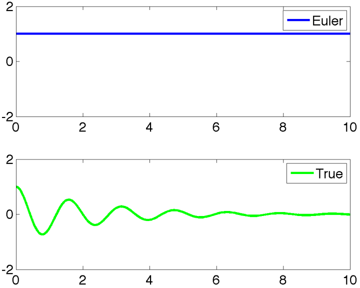


Verification of the equation solution (unforced case)



Something is wrong!
(numeric different from analytic)

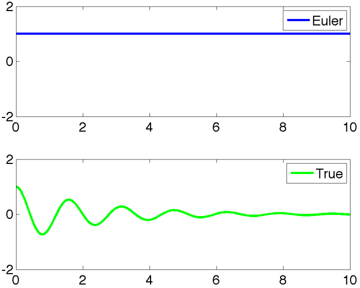
Verification of the equation solution (unforced case)



Something is wrong!
(numeric different from analytic)

There is a bug in euler.m routine:

Verification of the equation solution (unforced case)

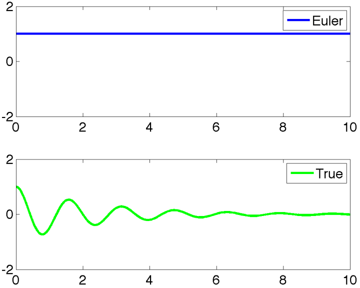


Something is wrong!
(numeric different from analytic)

There is a bug in euler.m routine:

```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi0);
```

Verification of the equation solution (unforced case)

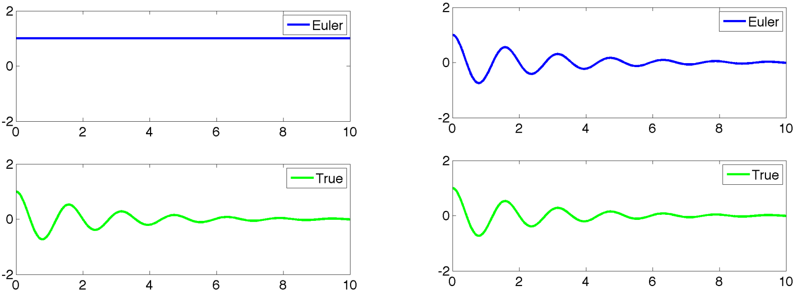


Something is wrong!
(numeric different from analytic)

There is a bug in euler.m routine:

```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi0);
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi(:,n));
```

Verification of the equation solution (unforced case)

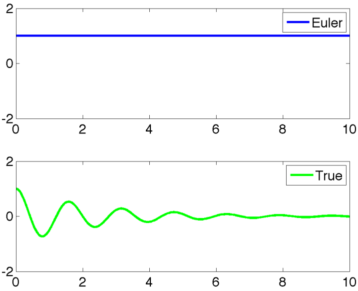


Something is wrong!
(numeric different from analytic)

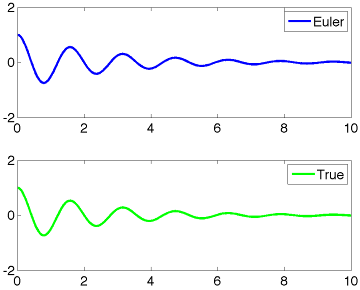
There is a bug in euler.m routine:

```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi0);
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi(:,n));
```

Verification of the equation solution (unforced case)



Something is wrong!
(numeric different from analytic)

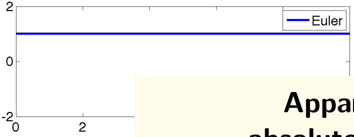


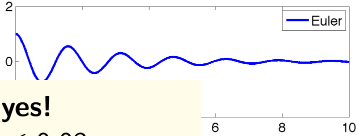
Is the code working fine?!

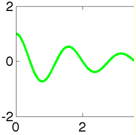
There is a bug in euler.m routine:

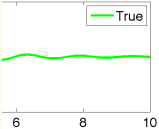
```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi0);
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi(:,n));
```

Verification of the equation solution (unforced case)



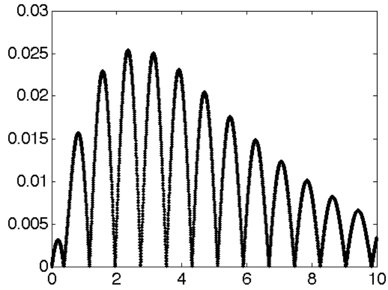






Apparently yes!

absolute error ≤ 0.03



Somethin ... **king fine?!**

(numeric differ

There is a bu

```
phi(:,n+1)
phi(:,n+1)    phi(:,n) + dt*rhs(t0,phi(:,n));
```

main_verification21.m

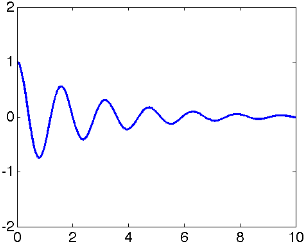
```
clc; clear all; close all;
m = 1.0; ksi = 0.1; wn = 4.0; f = 10.0; w = 5.0;
x0 = 1.0; v0 = 0.0; t0 = 0.0; t1 = 10.0; N = 3000;

dphidt=@(t,phi)[0 1; -wn^2 -2*ksi*wn]*phi ...
             + [0; (f/m)*sin(w*t)];

[time,phi] = euler(dphidt,[x0;v0],t0,t1,N);

plot(time,phi(1,:),'LineWidth',3);
```

Verification of the equation solution (forced case)

Imagine that analytical solution is not known.

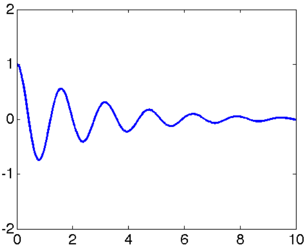
Verification of the equation solution (forced case)

Imagine that analytical solution is not known.



Verification of the equation solution (forced case)

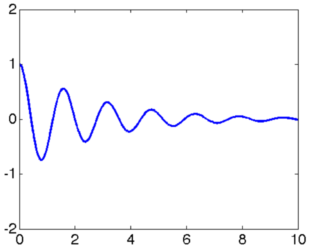
Imagine that analytical solution is not known.



Something is wrong!
(incompatible with harmonic forcing)

Verification of the equation solution (forced case)

Imagine that analytical solution is not known.

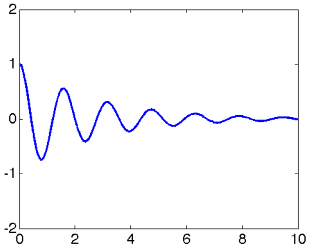


Something is wrong!
(incompatible with harmonic forcing)

The routine euler.m still has bug(s):

Verification of the equation solution (forced case)

Imagine that analytical solution is not known.



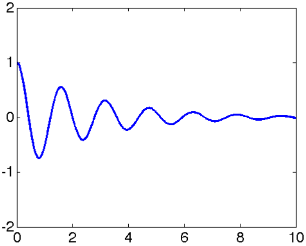
Something is wrong!
(incompatible with harmonic forcing)

The routine euler.m still has bug(s):

```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi(:,n));
```

Verification of the equation solution (forced case)

Imagine that analytical solution is not known.



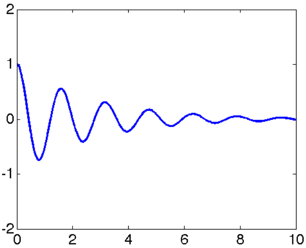
Something is wrong!
(incompatible with harmonic forcing)

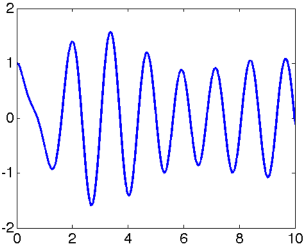
The routine euler.m still has bug(s):

```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi(:,n));
phi(:,n+1) = phi(:,n) + dt*rhs(time(1,n),phi(:,n));
```

Verification of the equation solution (forced case)

Imagine that analytical solution is not known.





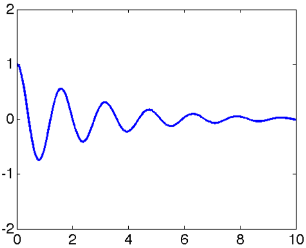
Something is wrong!
(incompatible with harmonic forcing)

The routine euler.m still has bug(s):

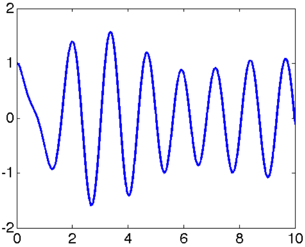
```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi(:,n));
phi(:,n+1) = phi(:,n) + dt*rhs(time(1,n),phi(:,n));
```

Verification of the equation solution (forced case)

Imagine that analytical solution is not known.



Something is wrong!
(incompatible with harmonic forcing)



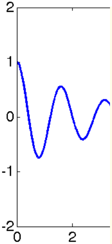
Is this the correct response?

The routine euler.m still has bug(s):

```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi(:,n));
phi(:,n+1) = phi(:,n) + dt*rhs(time(1,n),phi(:,n));
```

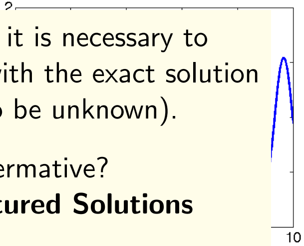
Verification of the equation solution (forced case)

Imagine that analytical solution is not known.



To answer this question it is necessary to compare numerical results with the exact solution (which is supposed to be unknown).

What is the alternative?
Method of Manufactured Solutions



Something is wrong!
(incompatible with harmonic forcing)

Is this the correct response?

The routine euler.m still has bug(s):

```
phi(:,n+1) = phi(:,n) + dt*rhs(t0,phi(:,n));
phi(:,n+1) = phi(:,n) + dt*rhs(time(1,n),phi(:,n));
```

Method of Manufactured Solutions

The idea is to construct (manufacture) an initial value problem (IVP) in which the solution is known, and use it to test the numerical integrator functionality.

1. Choose the form of model equations

$$\dot{\phi} = f\left(t, \phi\right), \quad \phi(0) = \phi_0 \quad (\star)$$

2. Define a manufactured solution Θ such that $\Theta(0) = \phi_0$, which does not verify $(\star)$, i.e. $\dot{\Theta} \neq f\left(t, \Theta\right)$.
3. Compute the residue function $\mathcal{R}(t) := \dot{\Theta} - f\left(t, \Theta\right) \neq 0$.
4. Define the manufactured IVP

$$\dot{\Theta} = f\left(t, \Theta\right) + \mathcal{R}(t), \quad \Theta(0) = \phi_0,$$

which is verifyed by the manufactured solution Θ.

Method of Manufactured Solutions

Example: (Forced MSD Oscillator)

Take as manufactured solution $\Theta = \begin{bmatrix} \cos t & \sin t \end{bmatrix}^T$, which satisfies the initial conditions.

This is not a solution for the forced oscillator, since

$$\underbrace{\begin{bmatrix} -\sin t \\ \cos t \end{bmatrix}}_{\dot{\Theta}} \neq \underbrace{\begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\,\xi\,\omega_n \end{bmatrix} \begin{bmatrix} \cos t \\ \sin t \end{bmatrix} + \begin{bmatrix} 0 \\ (f/m)\,\sin(\omega\,t) \end{bmatrix}}_{f(t,\Theta)}.$$

Then, the residual function is

$$\mathcal{R}(t) = \begin{bmatrix} -2\,\sin t \\ \cos t + 2\,\xi\,\omega_n\,\sin t + \omega_n^2\,\cos t - (f/m)\sin(\omega\,t) \end{bmatrix}.$$

Method of Manufactured Solutions

The manufactured initial value problem is

$$\begin{bmatrix} \dot{\Theta}_1 \\ \dot{\Theta}_2 \end{bmatrix} = \begin{bmatrix} \Theta_2 - 2\,\sin t \\ -\omega_n^2\,\Theta_1 - 2\,\xi\,\omega_n\,\Theta_2 + \cos t + 2\,\xi\,\omega_n\,\sin t + \omega_n^2\,\cos t \end{bmatrix},$$

where $\Theta_1(0) = 1$ and $\Theta_2 = 0$. Indeed,

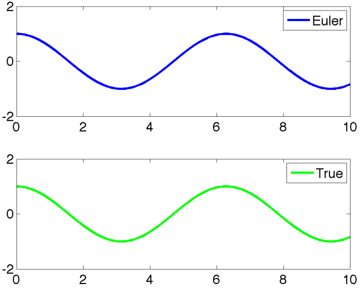
$$\begin{bmatrix} \Theta_1 \\ \Theta_2 \end{bmatrix} = \begin{bmatrix} \cos t \\ \sin t \end{bmatrix}$$

is a solution. Verify yourself!

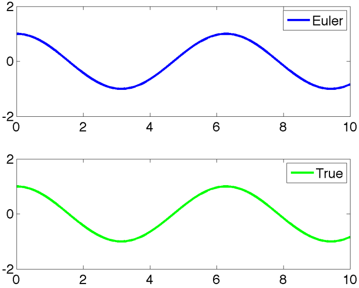
main_verification31.m

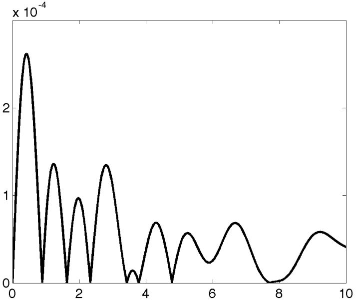
```
clc; clear all; close all;
m = 1.0; ksi = 0.1; wn = 4.0; f = 10.0; w = 5.0;
x0 = 1.0; v0 = 0.0; t0 = 0.0; t1 = 10.0; N = 5000;

dphidt=@(t,phi)[0 1; -wn^2 -2*ksi*wn]*phi ...
             + [-2*sin(t); ...
             cos(t)+2*ksi*wn*sin(t)+wn^2*cos(t)];

[time,phi] = euler(dphidt,[x0;v0],t0,t1,N);

x_true  = cos(time);

subplot(2,1,1)
plot(time,phi(1,:),'b','LineWidth',3);
legend('Euler')
subplot(2,1,2)
plot(time,x_true,  'g','LineWidth',3);
legend('True')
```

Verification of the equation solution (manufactured IVP)

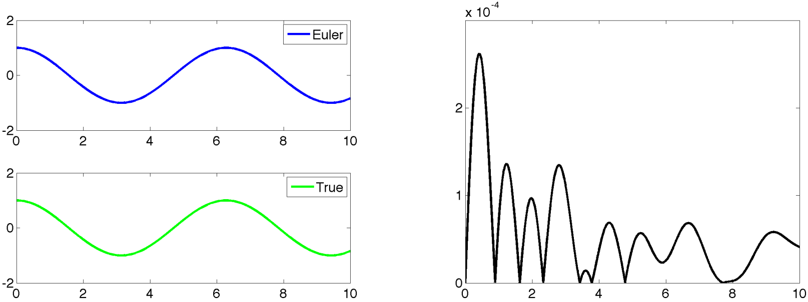


Verification of the equation solution (manufactured IVP)



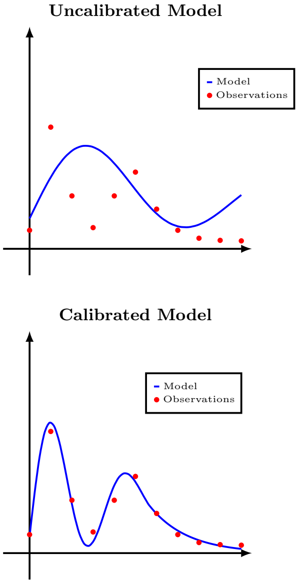


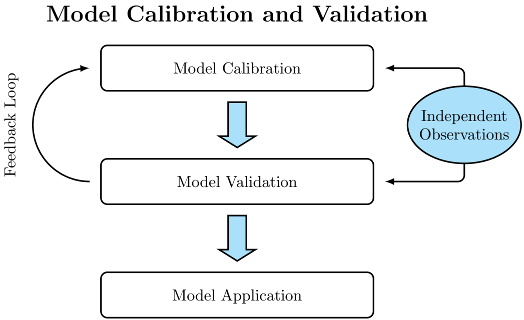
Verification of the equation solution (manufactured IVP)



Euler method is working fine!

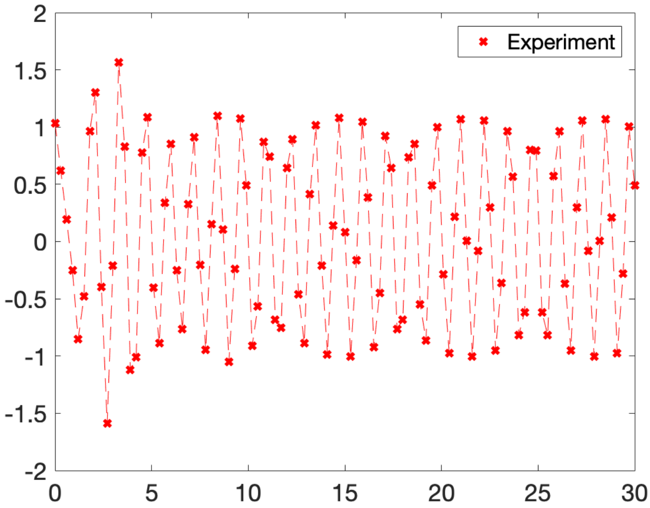
Model Calibration and Model Validation





`* Left pictures prepared by Michel Tosin.`

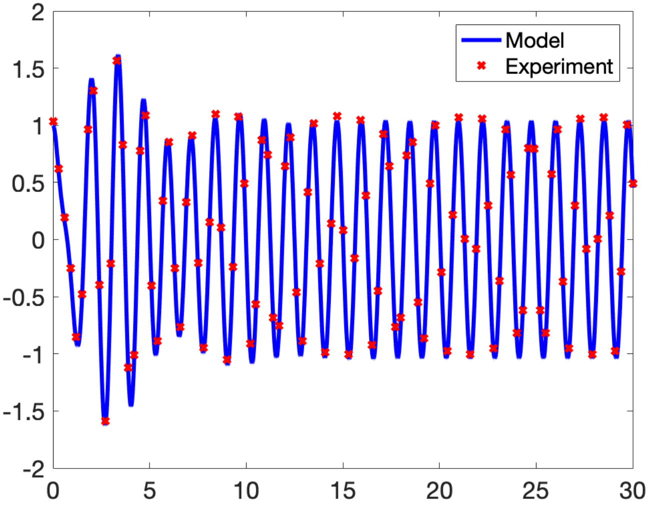
Validation case 1: experimental data set



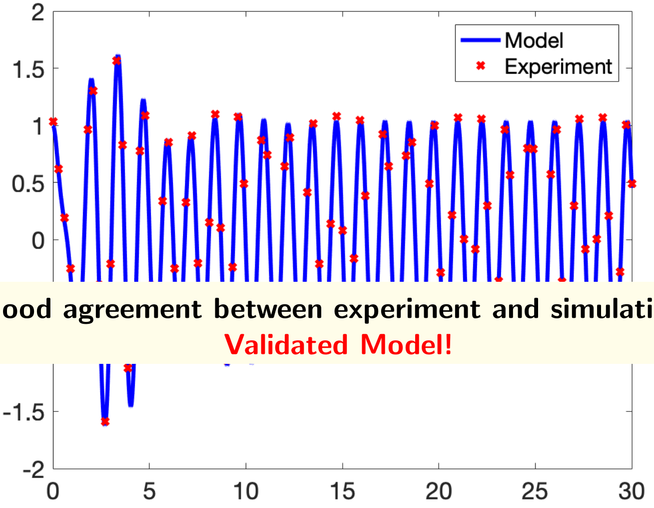
main_validation11.m

```
clc; clear; close all;
m = 1.0; ksi = 0.1; wn = 4.0; f = 10.0; w = 5.0;
x0 = 1.0; v0 = 0.0; t0 = 0.0; t1 = 30.0; N = 5000;

dphidt=@(t,phi)[0 1; -wn^2 -2*ksi*wn]*phi ...
             + [0; (f/m)*sin(w*t)];

[t,phi] = euler(dphidt,[x0;v0],t0,t1,N);

t_exp = t(1:50:end);
x_exp = phi(1,1:50:end) + 0.01*randn;

plot(t,phi(1,:),'b',t_exp,x_exp,'xr','LineWidth',3);
axis([t0 t1 -2 2])
legend('Model','Experiment')
```

Validation case 1: predictions and observations



Validation case 1: predictions and observations



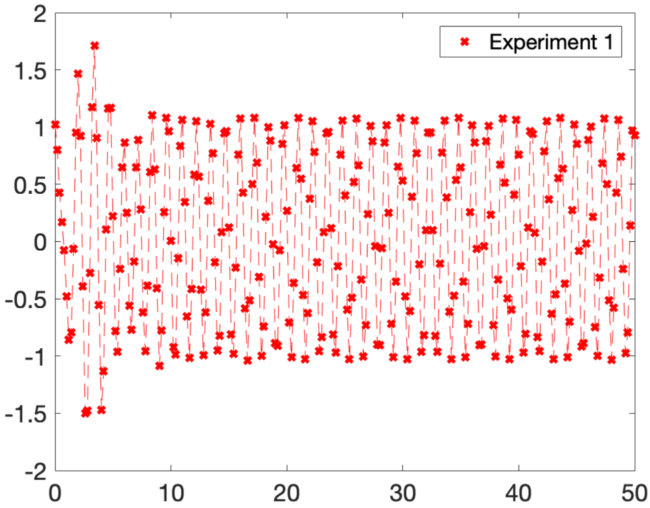
Good agreement between experiment and simulation!

Validated Model!

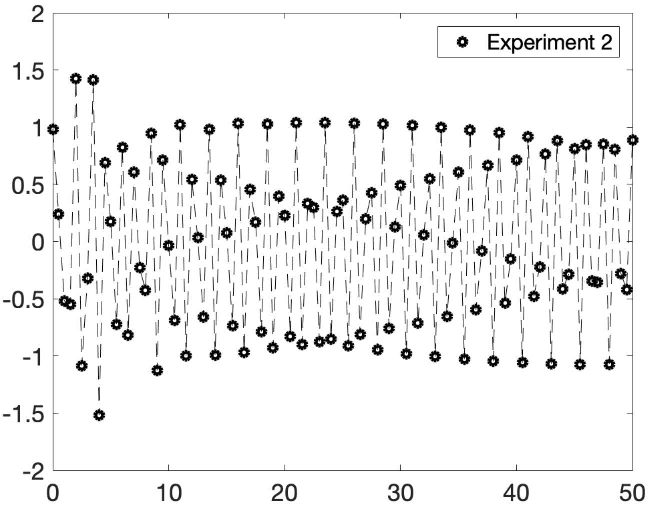
main_validation21.m (1/3)

```
clc; clear; close all;
m = 1.0; ksi = 0.1; wn = 4.0; f = 10.0; w = 5.0;
x0 = 1.0; v0 = 0.0; t0 = 0.0; t1 = 50.0; N = 5000;

dphidt=@(t,phi)[0 1; -wn^2 -2*ksi*wn]*phi + [0; (f/m)*sin(w*t)];
[t_ref,phi_ref] = euler(dphidt,[x0;v0],t0,t1,N);
t_exp1 = t_ref(1:20:end); x_exp1 = phi_ref(1,1:20:end) + 0.05*randn;
t_exp2 = t_ref(1:50:end); x_exp2 = phi_ref(1,1:50:end) + 0.1*randn;

figure(1)
plot(t_exp1,x_exp1,'xr','LineWidth',3);
hold on
plot(t_exp1,x_exp1,'--r','LineWidth',0.3);
hold off
axis([t0 t1 -2 2]); set(gca,'fontsize',18); legend('Experiment 1');

figure(2)
plot(t_exp2,x_exp2,'ok','LineWidth',3);
hold on
plot(t_exp2,x_exp2,'--k','LineWidth',0.3);
hold off
axis([t0 t1 -2 2]); set(gca,'fontsize',18); legend('Experiment 2');
```

Validation case 2: First experimental data set



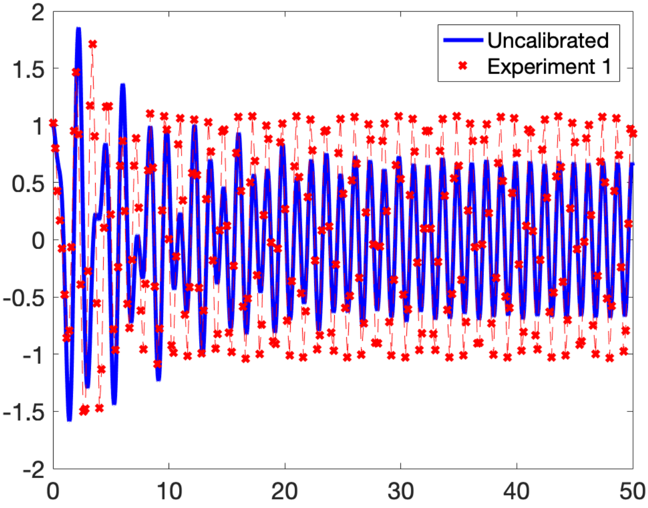
Validation case 2: Second experimental data set



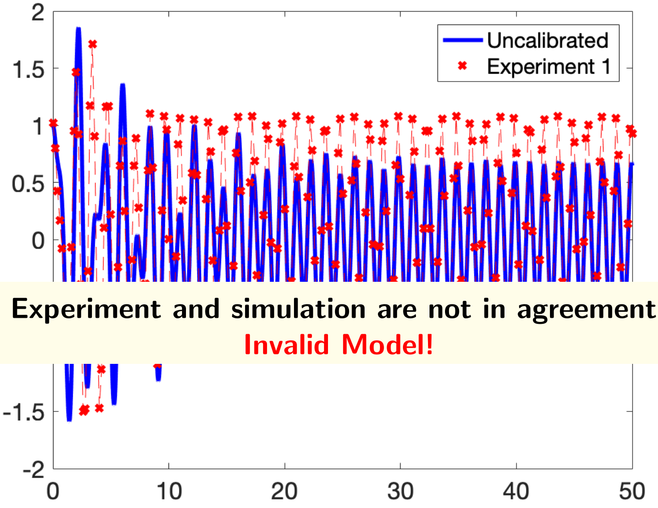
main_validation22.m (2/3)

```
m = 1.0; ksi = 0.05; wn = 3.2; f = 10.0; w = 5.0;
x0 = 1.0; v0 = 0.0;

dphidt=@(t,phi)[0 1; -wn^2 -2*ksi*wn]*phi + [0; (f/m)*sin(w*t)];
[t_uncal,phi_uncal] = euler(dphidt,[x0;v0],t0,t1,N);

figure(3)
plot(t_uncal,phi_uncal(1,:),'b','LineWidth',3);
hold on
plot(t_exp1,x_exp1,'xr','LineWidth',3);
plot(t_exp1,x_exp1,'--r','LineWidth',0.3);
hold off
axis([t0 t1 -2 2])
set(gca,'fontsize',18)
legend('Uncalibrated','Experiment 1')
```

Validation case 2: predictions and observations



Validation case 2: predictions and observations



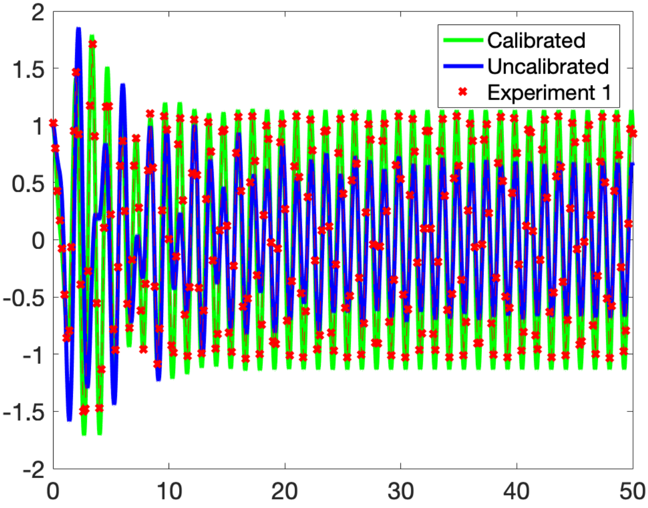
Experiment and simulation are not in agreement!

Invalid Model!

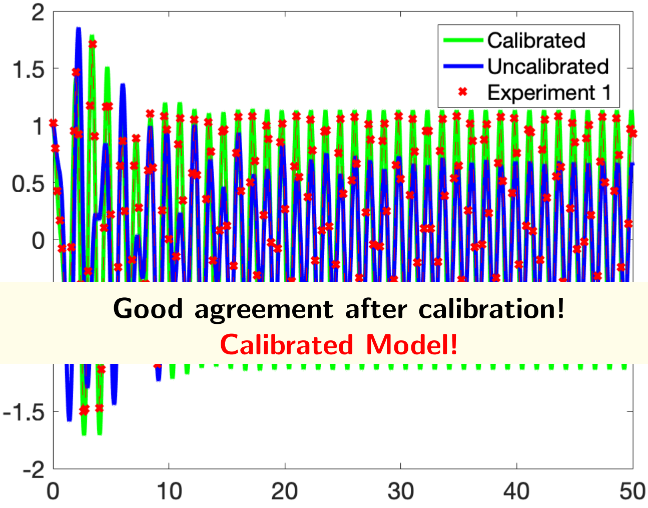
main_validation23.m (3/3)

```
m = 1.0; ksi = 0.095; wn = 4.08; f = 10.0; w = 5.0;
x0 = 1.0; v0 = 0.0;

dphidt=@(t,phi)[0 1; -wn^2 -2*ksi*wn]*phi + [0; (f/m)*sin(w*t)];
[t_cal,phi_cal] = euler3(dphidt,[x0;v0],t0,t1,N);

figure(4)
plot(t_cal,phi_cal(1,:),'g',t_uncal,phi_uncal(1,:),'b','LineWidth',3);
hold on
plot(t_exp1,x_exp1,'xr','LineWidth',3);
plot(t_exp1,x_exp1,'--r','LineWidth',0.3);
hold off
axis([t0 t1 -2 2]); set(gca,'fontsize',18);
legend('Calibrated','Uncalibrated','Experiment 1')

figure(5)
plot(t_cal,phi_cal(1,:),'g','LineWidth',3);
hold on
plot(t_exp1,x_exp1,'xr','LineWidth',3);
plot(t_exp2,x_exp2,'ok','LineWidth',3);
plot(t_exp1,x_exp1,'--r','LineWidth',0.3);
plot(t_exp2,x_exp2,'--k','LineWidth',0.3);
hold off
axis([t0 t1 -2 2]); set(gca,'fontsize',18);
legend('Calibrated','Experiment 1','Experiment 2')
```

Validation case 2: model calibration



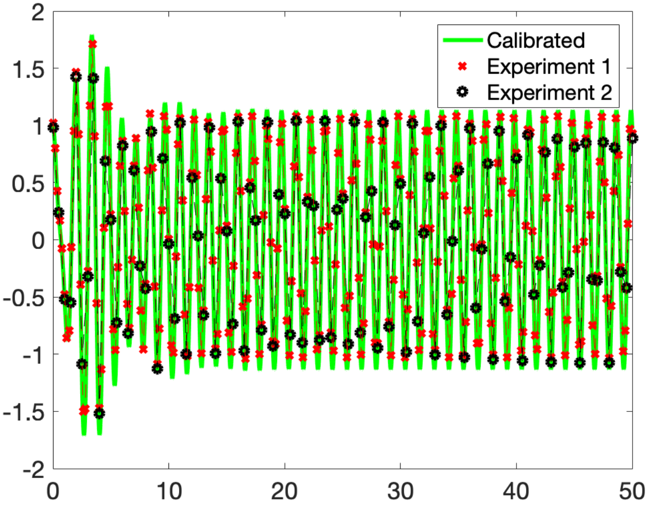
Validation case 2: model calibration



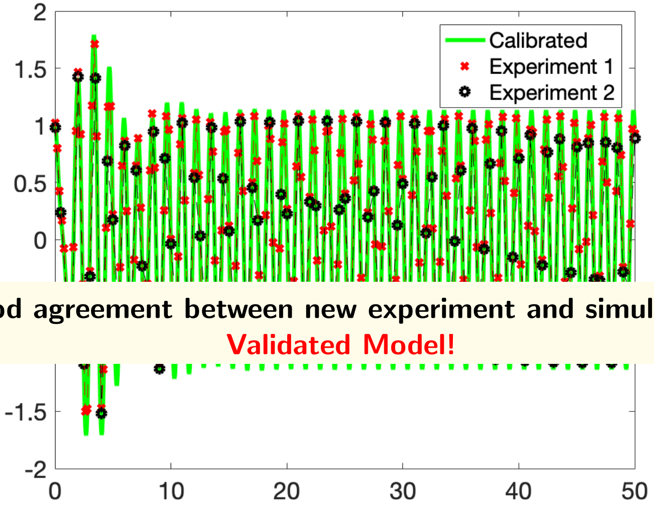
Good agreement after calibration!

Calibrated Model!

Validation case 2: calibrated model and new observations



Validation case 2: calibrated model and new observations



Good agreement between new experiment and simulation!
Validated Model!

References

C. J. Roy and W. L. Oberkampf, *A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing.* **Computer Methods in Applied Mechanics and Engineering**, 200: 2131–2144, 2011.

C. J. Roy, *Review of code and solution verification procedures for computational simulation.* **Journal of Computational Physics**, 205: 131–156, 2005.

W. L. Oberkampf, T. Trucano and C. Hirsch, *Verification, validation, and predictive capability in computational engineering and physics.* **Applied Mechanics Reviews**, 57: 345–384, 2004.

W. L. Oberkampf and C. J. Roy, **Verification and Validation in Scientific Computing**. Cambridge University Press, 1st edition, 2010.