

Method of Manufactured Solutions: Demonstrations

Len Schwer

Len@Schwer.net

August 2002

Introduction	2
Model Problem: Spring and Mass System	3
Analytical Solution	3
Numerical Solution	4
Verification of the Numerical Solution	5
L_2 Norm Error Measure	5
Observed Order-of-Accuracy	6
Convergence Study	7
MMS Demonstration with Intentional Errors	9
Three Intentional Errors	9
Error 1: Sign Error in Initial Condition	9
Error 2: Typographical Error of a Constant	10
Error 3: Algorithmic Development Error	10
Case 1 Manufactured Solution: $x(t) = A \sin \omega t + B \cos \omega t$	10
Case 2 Manufactured Solution: $x(t) = 1 + A \exp(B\omega t)$	12
Conclusions	14
Recommendations	15
Acknowledgements	16
References	16
Appendix – Microsoft Excel VBA Macro for Spring-Mass MMS Demonstration	17

Introduction

Verifying numerical algorithms for all of their intended uses is a daunting (impossible?) task. There is a vast middle ground between a code developer who checks a routine for a few very special cases, and the user, who believing the code is verified, uses the algorithm in a typical large scale application, which exercises many (untested) possible paths and combinations of parameters, but essentially without the ability to identify any minor algorithmic or coding errors in the large amount of results. The Method of Manufactured Solutions (MMS) attempts to bridge this wide gap between verifying a few special cases and fully exercising a routine with the aim of identifying the existence of minor algorithmic or coding errors.

Roache (2002) provides this assessment of MMS:

“Verification of Calculations involves error estimation, whereas Verification of Codes involves error evaluation, from known benchmark solutions. The best benchmarks are exact analytical solutions with sufficiently complex solution structure; they need not be realistic since Verification is a purely mathematical exercise. The Method of Manufactured Solutions (MMS) provides a straightforward and quite general procedure for generating such solutions. For complex codes, the method utilizes Symbolic Manipulation, but here it is illustrated with simple examples. When used with systematic grid refinement studies, which are remarkably sensitive, MMS produces strong Code Verifications with a theorem-like quality and a clearly defined completion point.”

In the typical solution of differential equations, the givens are the differential equations, corresponding forcing functions, i.e. so called ‘right hand sides’, and the corresponding initial and boundary conditions. The goal is to determine the analytical solution, consistent with these givens. In using MMS, a numerical solution algorithm already exists, which should treat arbitrary initial and boundary conditions, and forcing functions, in a consistent, and general, manner. The MMS allows for the selection of an arbitrary analytical solution, *without* regard to the differential equation. This manufactured solution is used to determine the consistent boundary and initial conditions, and via substitution into the differential equation, the determination of the consistent source term. It is the application of these consistent initial and boundary conditions, and forcing function, to the numerical solution algorithm that provides the verification of the numerical solution algorithm by demonstrating that the stated order-of-accuracy of the solution algorithm is implemented correctly in the program.

The order-of-accuracy criteria is key to the effectiveness of the MMS. The direct use of the more traditional error measures, i.e. relative error, are inadequate. Subsequent grid refinement will (almost) always show an *eventual* convergence of the numerical algorithm. By comparing the relative error from only two grid refinements, i.e. the original grid and one refinement, the order-of-accuracy of the numerical algorithm can be estimated and compared to the theoretical order-of-accuracy of the algorithm. This comparison is very sensitive to all errors in the numerical algorithm, and can also detect errors in the derivation of the algorithm.

Model Problem: Spring and Mass System

To illustrate the basic application of the Method of Manufactures Solutions (MMS) a simple model problem is constructed which is easily discretized for numerical solution using the central difference operator. The analytical solution for the model problem is well known. Although not necessary when using the MMS, in this demonstration the analytical solution serves to introduce the reader to the MMS using a familiar analytical solution in the role of a manufactured solution.

The model problem is a second-order ordinary differential equation which describes the motion of an undamped spring-mass system

$$\ddot{x} + \omega^2 x = F(t) \quad (1)$$

where x is the displacement of the mass, the superscript double dots indicate double differentiation with respect to time t , ω is the natural frequency of the system, and $F(t)$ is the time varying forcing function. The complete problem specification requires two initial conditions:

$$\begin{aligned} x(0) &= \alpha \\ \dot{x}(0) &= \beta \end{aligned} \quad (2)$$

where α and β are constants.

Analytical Solution

The general solution to Equation (1) is given by

$$x(t) = A \sin \omega t + B \cos \omega t + (1 - \omega^2) \iint [F(t) dt] dt \quad (3)$$

Without loss of generality, for the present purpose, consider the case where $F(t) = 0$, i.e. free vibrations

$$\ddot{x} + \omega^2 x = 0 \quad (4)$$

then the general solution of Equation (4) is given by

$$x(t) = A \sin \omega t + B \cos \omega t \quad (5)$$

and from the initial conditions Equations (2)

$$\begin{aligned}x(0) &= \alpha = B \\ \dot{x}(0) &= \beta = A\omega\end{aligned}\tag{6}$$

Thus the analytical solution can be written as

$$x(t) = \frac{\beta}{\omega} \sin \omega t + \alpha \cos \omega t\tag{7}$$

Numerical Solution

Equation (1) can be written in a discrete form as

$$\ddot{x}_n + \omega^2 x_n = F_n\tag{8}$$

where the subscript n is a notation for the discrete time, i.e.

$$x_n = x(t_n) = x(n\Delta t)\tag{9}$$

and Δt is the time increment.

The numerical solution of Equation (8) is obtained using the central difference operator,

$$\dot{x}_{n+1/2} = \frac{x_{n+1} - x_n}{\Delta t}\tag{10}$$

The velocity in Equation (10) is referred to as the mid-point velocity as it is defined by the end points of the time increment. In a similar manner the acceleration can be written as

$$\ddot{x}_n = \frac{\dot{x}_{n+1/2} - \dot{x}_{n-1/2}}{\Delta t} = \frac{x_{n+1} - 2x_n + x_{n-1}}{\Delta t^2}\tag{11}$$

Applying the above central difference operator to Equation (8) yields

$$x_{n+1} = \Delta t^2 F_n + \left[2 - (\Delta t \omega)^2 \right] x_n - x_{n-1}\tag{12}$$

To complete the numerical solution, we consider the initial conditions

$$\begin{aligned}x(0) &= x_0 = \alpha \\ \dot{x}(0) &= \dot{x}_0 = \frac{x_{1/2} - x_{-1/2}}{\Delta t} = \frac{x_1 - x_{-1}}{2\Delta t} = \beta\end{aligned}\tag{13}$$

where the time increment index in the second of Equations (13) was shifted using the following relations

$$x_{1/2} = \frac{x_1 + x_0}{2} \quad \text{and} \quad x_{-1/2} = \frac{x_0 + x_{-1}}{2} \quad (14)$$

Thus the second of Equations (13), i.e. the velocity initial condition, can be written as

$$x_{-1} = x_1 - 2\beta\Delta t \quad (15)$$

The values of x_{-1} and x_1 are obtained by solving Equation (15) and Equation (12), evaluated at $n = 0$, to obtain the following expression for x_1

$$x_1 = \frac{1}{2} \left\{ \Delta t (F_0 \Delta t + 2\beta) + \left[2 - (\omega \Delta t)^2 \right] x_0 \right\} \quad (16)$$

Thus the numerical solution given by Equation (12) starts with the evaluation of x_2 .

Verification of the Numerical Solution

Before *intentional* errors are introduced into the numerical solution algorithm, to demonstrate the Method of Manufactured Solutions, we seek to establish that the *as programmed* algorithm is correct. The traditional method for verifying such an algorithm is to compare the analytical solution, typically of a relatively simple problem, with the corresponding numerical solution. This is precisely the essence of the Method of Manufactured Solutions, however, more complex solutions are typically used, to thoroughly exercise the programmed algorithm.

The comparison between the analytical solution and the numerical results are typically made via an error measure. Essentially any error measure will serve this purpose, but perhaps the most widely used is the L_2 norm.

L_2 Norm Error Measure

Salari and Knupp (2000) provide the following description of the L_2 error norm as it relates to the Method of Manufactured Solutions. For discrete functions U and V the l_2 norm¹ of $U - V$ is

¹ The lower case l_2 norm is used to emphasize discrete solutions with summations over points, as opposed to continuous solutions and corresponding spatial integration denoted traditionally by upper case L_2

$$|U - V|_2 = \sqrt{\sum_n (U - V)^2 \eta_n} \quad (17)$$

where η is some local volume measure and n is the index of the discrete solution location, or time.

As will be shown, the manufactured solution is a continuum solution, and thus can be evaluated at the same time and space locations as the discrete solution. The local error at a point n in the grid is given by the difference between the manufactured solution u_n and the discrete solution U_n . The normalized global error is defined by

$$e_2 = \sqrt{\frac{\sum_n (u_n - U_n)^2 \eta_n}{\sum_n \eta_n}} \quad (18)$$

If the local volume measure is constant, i.e. uniform grid, the normalized global error, sometimes referred to as the Root Mean Square (RMS) error, is defined as

$$e_2 = \sqrt{\frac{1}{N} \sum_n (u_n - U_n)^2} \quad (19)$$

Observed Order-of-Accuracy

Salari and Knupp (2000) provide the following useful discussion on estimating the order-of-accuracy by comparing the measured error for successively refined grids. A more in-depth discussion is presented by Roache (1998), and distinctions for the terminology of theoretical, actual asymptotic, and observed order-of-accuracy are given by Westerink and Roache (1995).

The definition of the theoretical order-of-accuracy is related to the discretization error and is a function of h , where h is the grid spacing, or similarly the time increment:

$$E = E(h) \quad (20)$$

For consistent methods, the discretization error E in the solution is proportional to h^p where $p > 0$ is the theoretical order of the method, i.e.

$$E = Ch^p + HOT \quad (21)$$

where C is a constant and HOT refers to higher order terms. If the numerical procedure is consistent, then the continuum equations represented by the numerical procedure should be recovered as h goes to zero.

To estimate the order-of-accuracy of the code being verified, the assumption is made that as h goes to zero, the first term in Equation (21) dominates. The global errors E_{grid1} and E_{grid2} are obtained from the numerical solutions to the manufactured solution test problem at two grid refinements. If $grid1$ is of size h and $grid2$ is of size h/r , where r is the refinement ratio, then the errors have the form

$$\begin{aligned} E_{grid1} &\approx Ch^p \\ E_{grid2} &\approx C\left(\frac{h}{r}\right)^p \end{aligned} \quad (22)$$

The ratio of these errors is given by

$$\frac{E_{grid1}}{E_{grid2}} \approx \left(\frac{Ch^p}{Ch^p}\right)r^p = r^p \quad (23)$$

Thus the observed order p is computed as

$$p = \frac{1}{\ln r} \ln \left(\frac{E_{grid1}}{E_{grid2}} \right) \quad (24)$$

Systematic grid refinement, or time step refinement, with a constant refinement ratio r produces a sequence of observed orders-of-accuracy. The trend in the sequence is then compared to the theoretical order-of-accuracy of the discrete method. Typically the theoretical order-of-accuracy is not recovered to more than two, or rarely three, significant figures since as h goes to zero the machine precision errors become significant. Equation (24) is also applicable to assessing the accuracy of the time domain discretization where h is replaced by Δt .

Convergence Study

The central difference operator, used to solve Equation (8), is second order accurate, see for example Belytschko, et al. (2000, page 316). Thus in verifying the numerical implementation of the above described solution algorithm, the algorithm is given in Appendix I, we expect to see an order-of-accuracy value of $p = 2$.

The data for the convergence study is:

$$\begin{aligned}
 x(0) &= \alpha = 0 \\
 \dot{x}(0) &= \beta = 1 \\
 \omega &= 2\pi \\
 \Delta t &= t_f / n
 \end{aligned}
 \tag{25}$$

with a final time (duration) of $t_f = 1.1$ and n is the total number of steps listed; note 20 uniformly spaced time-samples of the analytical and numerical results were used to determine the RMS, i.e. N in Equation (19). Figure 1 shows a comparison of the analytical solution with the numerical solution where a total of 40 time steps were used in the solution, and the 20 output times are indicated by the data symbols. Table 1 summaries the results of six time increment refinement studies. The RMS value in this table is obtained from Equation (19), and the order-of-accuracy p is obtained from Equation (24). This convergence study verifies that the solution algorithm, as implemented, is second order accurate as expected.

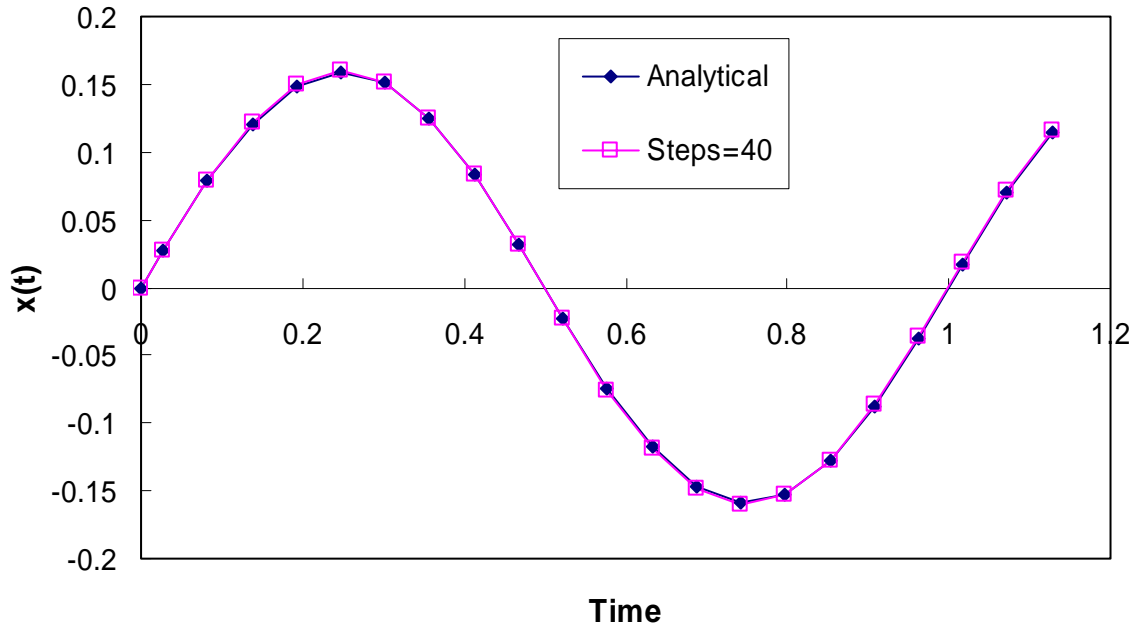


Figure 1 Comparison of analytical and numerical (40 time steps) solution of spring-mass system used for convergence study.

Table 1 Summary of convergence study for solution of spring-mass system.

Nsteps	RMS	p
20	3.18E-03	2.065
40	7.60E-04	2.032
80	1.86E-04	2.016
160	4.59E-05	2.008
320	1.14E-05	2.004
640	2.85E-06	

MMS Demonstration with Intentional Errors

Using the above described verified numerical algorithm for the spring-mass problem, in this section a demonstration of the Method of Manufactured Solutions (MMS) is presented by intentionally introducing errors in the algorithm. As a further demonstration that *any* analytical solution can be used in the MMS, two analytical solution cases are considered:

1. $x(t) = A \sin(\omega t) + B \cos(\omega t)$
2. $x(t) = A [1 + \exp(B\omega t)]$

with the same initial conditions given above as Equation (2). Case 1 is a continuation of the above numerical example. In addition to demonstrating the MMS via order-of-accuracy studies, it provides a basis for assessing the effects of the intentional errors on the solution. Case 2 should be considered a more typical application of MMS, as often for complex systems of partial differential equations, known solutions, such as Case 1, are not available, and are *not* required by the MMS.

Three Intentional Errors

No claim is made that the selected errors are the best, or worst, but rather only plausible errors, i.e. that might occur in deriving and implementing the numerical solution of a simple spring-mass system. The Appendix includes a listing of the Microsoft Excel VBA macro where the three errors are optionally included in the program using IF-type statements.

Error 1: Sign Error in Initial Condition

As derived above, the correct initial condition specified for the first derivative is given by Equation (16), and repeated here

$$x_1 = \frac{1}{2} \left\{ \Delta t (F_0 \Delta t + 2\beta) + \left[2 - (\omega \Delta t)^2 \right] x_0 \right\} \quad (26)$$

Error 1 introduces the a sign mistake in the first term of this expression, viz.

$$x_1 = \frac{1}{2} \left\{ \Delta t (F_0 \Delta t - 2\beta) + \left[2 - (\omega \Delta t)^2 \right] x_0 \right\} \quad (27)$$

This corresponds to a negative initial velocity specification when $F_0 = 0$, since x_0 is zero, see Equations (2).

Error 2: Typographical Error of a Constant

The central difference numerical solution is given above by Equation (12) and is repeated here

$$x_{n+1} = \Delta t^2 F_n + \left[2 - (\Delta t \omega)^2 \right] x_n - x_{n-1} \quad (28)$$

Error 2 corresponds to changing the number 2 inside the brackets to the number 1, i.e.

$$x_{n+1} = \Delta t^2 F_n + \left[1 - (\Delta t \omega)^2 \right] x_n - x_{n-1} \quad (29)$$

Error 3: Algorithmic Development Error

Error 3 corresponds to an error in the algorithm derivation, as opposed to a coding error. The central difference numerical solution given above by Equation (12) is replaced with the following:

$$x_{n+1} = \frac{\Delta t^2 F_n + 2x_n - x_{n-1}}{1 + (\Delta t \omega)^2} \quad (30)$$

The above difference equation ‘solution’ was erroneously obtained by the author during the initial development of the examples in this manuscript. Rather than burying this mistake, it was decided to use this error as an illustration. As will be shown, this particular error demonstrates how an incorrect solution can appear to be *nearly* correct, but the Method of Manufactured Solutions readily identifies the existence of an error in the numerical solution.

Case 1 Manufactured Solution: $x(t) = A \sin \omega t + B \cos \omega t$

It has already been demonstrated that substitution of this analytical solution into the original differential equation, Equation (1), satisfies the differential equation when $F(t) = 0$. Table 2 summarizes the order-of-accuracy results for the error-free solution and the above defined three intentional errors, where only 20 and 40 time steps were used to assess the rate-of-convergence parameter p .

Table 2 Summary of Case 1 order-of-accuracy study.

Error	n	e_2	p
0	20	0.0008	
	40	0.0032	2.0651
1	20	0.2186	
	40	0.2130	0.0370
2	20	0.1185	
	40	0.1092	0.1184
3	20	0.0456	
	40	0.0274	0.7360

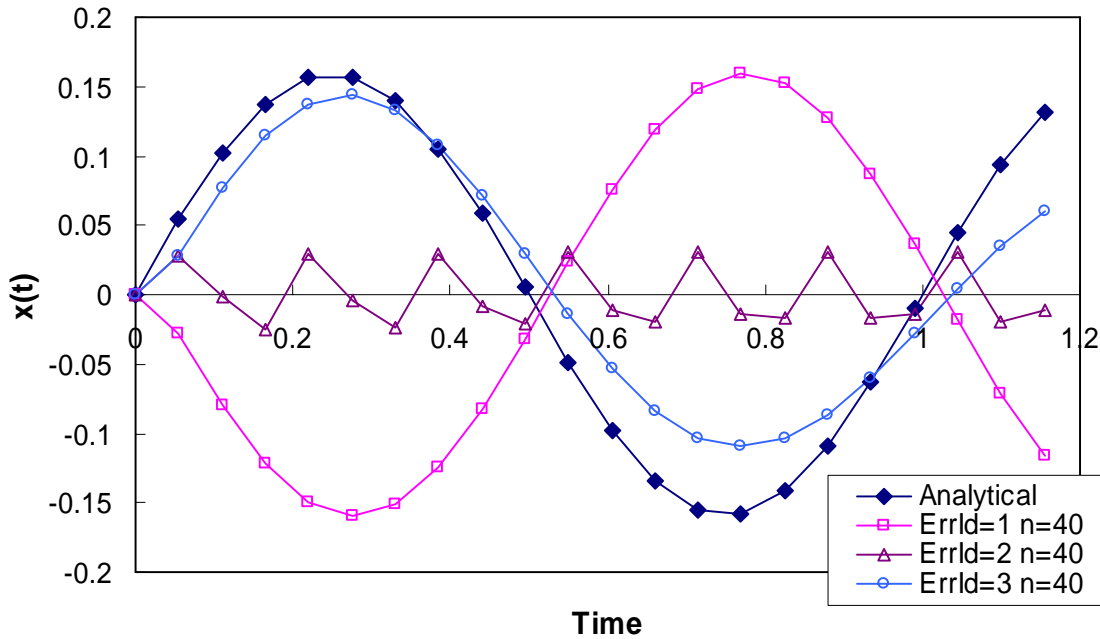


Figure 2 Illustration of Case 1 analytical solution with solutions corresponding to three intentional errors.

Figure 2 compares the analytical solution with the numerical solutions generated by the three intentional errors described above. Perhaps the most interesting of the three intentional errors is Error 3 where the central difference solution algorithm was incorrectly derived. Of the three intentional errors, the solution for Error 3 looks the most like the analytical solution. A careful analyst should recognize that the other two error solutions as indicating something is wrong with the algorithm, but the Error 3 solution might easily escape detection. Although the Error 3 solution may look quite similar to the expected solution, as indicated in Table 2, the rate-of-convergence for Error 3 is less than half of the expected order-of-convergence, thus indicating an error in the algorithm.

Case 2 Manufactured Solution: $x(t) = 1 + A \exp(B\omega t)$

This case provides a more typical application of the Method of Manufactured Solutions, as often, especially for complex systems of partial differential equations typical of mechanics applications, known solutions, such as Case 1, are not available.

As stated in the Introduction, in the typical solution of differential equations the givens are the differential equations, corresponding forcing functions, and the corresponding initial and boundary conditions. The goal is to determine the analytical solution, consistent with these givens. In using the Method of Manufactured Solutions, a numerical solution algorithm already exists, which should treat arbitrary initial and boundary conditions, and forcing functions, in a consistent, and general, manner. The MMS allows for the selection of an arbitrary analytical solution, *without* regard to the differential equation. This analytical solution is used to determine the consistent boundary and initial conditions, and via substitution into the differential equation, the determination of the consistent source term. It is the application of these consistent initial and boundary conditions, and source term, to the existing numerical solution algorithm that provides the verification of the numerical solution algorithm by demonstrating the stated order-of-accuracy of the solution algorithm.

The manufactured solution

$$X(t) = 1 + A \exp(B\omega t) \quad (31)$$

is applied to the differential equation, Equation (1), written here in operator notation so as to follow description by Roache (2002),

$$L(x) = \ddot{x} + \omega^2 x - F(t) = 0 \quad (32)$$

to determine the source term $Q(t)$, which when added to the differential equation produces the solution $x(t) = X(t)$, i.e.

$$Q(t) = L[X(t)] \quad (33)$$

By elementary operations on the manufactured solution $U(x)$, the following source term is obtained

$$Q(t) = \omega^2 [1 + A(B^2 + 1) \exp(B\omega t)] - F(t) \quad (34)$$

Thus the verification seeks to solve the modified equation

$$L(x) = \ddot{x} + \omega^2 x - F(t) = Q(t) \quad (35)$$

with compatible initial and boundary conditions, where the exact solution is given by $X(t)$ as specified in Equation (31).

The initial and boundary conditions can also be manufactured, but must be consistent with the differential equation and the manufactured solution. For the present demonstration, the same initial conditions, Equations (2), are maintained as specified for Case 1, and the analytical solution. The corresponding manufactured boundary conditions are

$$\begin{aligned} x(0) = \alpha = 1 + A &\Rightarrow A = \alpha - 1 \\ \dot{x}(0) = \beta = AB\omega &\Rightarrow B = \frac{\beta}{\omega(\alpha - 1)} \end{aligned} \tag{36}$$

Table 3 summarizes the order-of-accuracy results for the error-free solution and the above defined three intentional errors, where only 20 and 40 time steps were used to assess the rate-of-convergence.

Table 3 Summary of Case 2 order-of-accuracy study.

Error	n	e_2	p
0	20	0.0000	
	40	0.0001	2.0478
1	20	0.2135	
	40	0.2205	0.0468
2	20	0.4300	
	40	0.4093	-0.0712
3	20	0.0205	
	40	0.0369	0.8504

Figure 3 compares the analytical solution with the numerical solutions generated by the error-free algorithm, and with the three intentional errors described above. Again the most interesting of the three intentional errors is Error 3 where the central difference solution algorithm was incorrectly derived. The order-of-accuracy criteria seems to be quite sensitive to all three types of intentional errors.

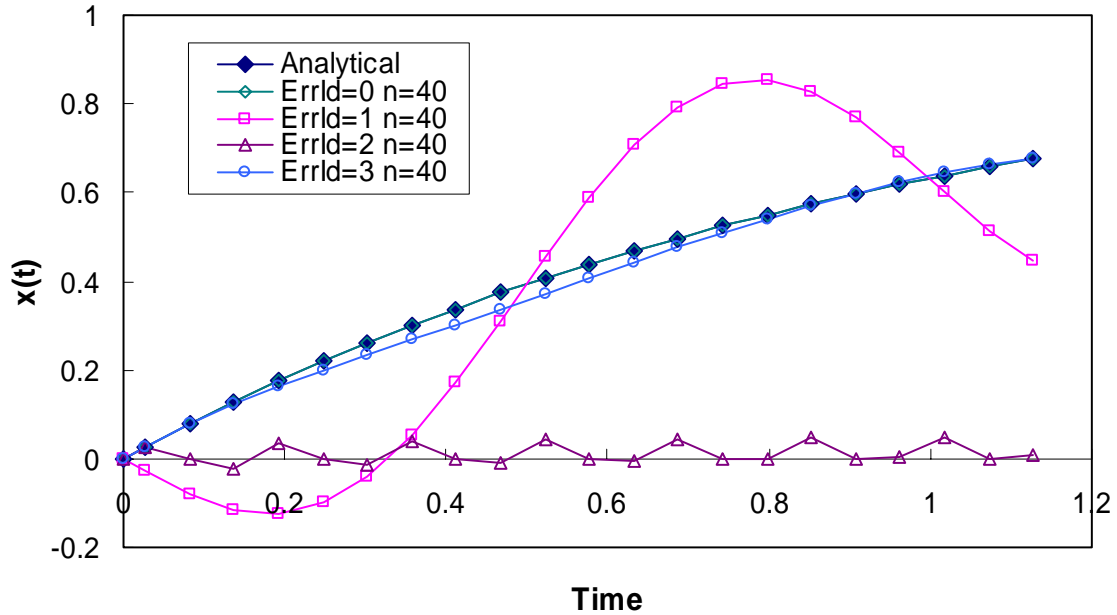


Figure 3 Illustration of Case 2 analytical solution with solutions corresponding to three intentional errors.

Conclusions

This demonstration of the Method of Manufactured Solutions provides a detailed description of how the method works with two case studies indicating the performance of the method. The author is now more impressed with the method, after working through the two “error free” cases and discovering numerous programming and derivation errors for even this most simple numerical algorithm.

Most² of the applications of MMS have been in the area of computational fluid dynamics, where accuracy and consistency have a greater emphasis than in solid mechanics. A possible downside to the application of the Method of Manufactured Solutions (MMS), in solid mechanics, is determining the theoretical order-of-accuracy of the algorithm to be verified. A routine may contain different types of discrete approximations, that operate on different variables, in an effort to obtain computational efficiency at the expense of consistency. However, the application of MMS would, at a minimum, provide the (lowest) order-of-accuracy, and perhaps motivate an investigation of how the convergence could be improved (corrected).

² An early application, of essentially MMS, to a plate bending problem is presented by Batra and Liang (1997).

Recommendations

One area of computational solid mechanics that seems most appropriate for the Method of Manufactured Solutions is verification of constitutive models. These material models can be verified as stand-alone program units using strain increment driving utilities, i.e. Material Model Drivers. Further, most of the material models are based on the same system of differential equations and thus the same suite of manufactured solutions can be applied to a number of constitutive models.

The basic differential equations to be solved are Hooke' Law

$$\dot{\sigma}_{ij} = C_{ijkl} (\dot{\varepsilon}_{ij} - \dot{\varepsilon}_{ij}^p) \quad (37)$$

where the superscript dot indicates either a rate with respect to time, or load. Here the stress tensor is σ_{ij} , the total strain is ε_{ij} , the plastic portion of the strain is ε_{ij}^p , and the elastic constitutive tensor is C_{ijkl} . The total strain is to be specified, i.e. manufactured, and the plastic strains are determined from a given plastic potential function G

$$\varepsilon_{ij}^p = \lambda \frac{\partial G}{\partial \sigma_{ij}} \quad (38)$$

For associated flow, the plastic potential function is identical to the yield function, i.e. $G \equiv f$, where the yield function specifies the stress space boundary between elastic and plastic response.

The consistency parameter $\dot{\lambda}$ is obtained from the consistency condition, i.e. the plastic stress state must be on the yield surface, which is expressed mathematically as

$$\dot{f} = \frac{\partial f}{\partial \sigma_{ij}} \dot{\sigma}_{ij} + \frac{\partial f}{\partial q} \dot{q} = 0 \quad (39)$$

where q represents other state variables, e.g. a hardening parameter, which are assumed to have an evolution of the form

$$\dot{q} = \dot{\lambda} H_q(\sigma, q) \quad (40)$$

The expression for $\dot{\lambda}$ is obtained by substituting Equations (38), (37), and (40) into Equation (39) and solving for $\dot{\lambda}$

$$\dot{\lambda} = \frac{(\partial_{\sigma_{ij}} f) C_{ijkl} \dot{\varepsilon}_{kl}}{(\partial_{\sigma_{ij}} f) C_{ijkl} (\partial_{\sigma_{kl}} f) - (\partial_q f) H_q} \quad (41)$$

where the compact notation

$$\partial_{\sigma_{ij}} f = \frac{\partial f}{\partial \sigma_{ij}} \quad (42)$$

has been adopted.

Solution of the above general plasticity algorithm is then straightforward as once the total strain rates are specified (manufactured) and the yield f and hardening functions H are given, then $\dot{\lambda}$ is obtained directly from Equation (42). This in turn provides the plastic strain rates from Equation (38), and lastly the stress rates are obtained from Equation (37).

The Method of Manufactured Solutions would thus seek to determine the order-of-accuracy of the numerical algorithm used to solve for the stress rates, i.e. the constitutive model routine.

Acknowledgements

The author gratefully acknowledges the support of Sandia National Laboratories under Contract PO-29753 and associated technical direction of Drs. Rebecca Brannon and William Scherzinger. The author is particularly grateful for the constructive comments of Dr. Patrick Roache.

References

- Batra, R.C. and X.Q. Liang, "Finite Dynamic Deformations of Smart Structures," *Computational Mechanics*, Volume 20, pages 427-438, 1997.
- Belytschko, T.B., W.K. Liu, and B. Moran, "Nonlinear Finite Elements for Continua and Structures," John Wiley & Sons Ltd., New York, 2000.
- Roache, P.J., "Verification and Validation in Computational Engineering," Hermosa Publishers, Albuquerque, NM, ISBN-0-913478-08-3, 1998.
- Roache, P.J., "Code Verification by the Method of Manufactured Solutions," *ASME Journal of Fluids Engineering*, Volume 124, Issue 1, pages 4-10, March, 2002
- Salari, K. And P. Knupp, "Code Verification by the Method of Manufactured Solutions," Sandia National Laboratories, SAND2000-1444, June, 2000.
- Westerink, J.J. and Roache, P.J. "Issues in Convergence Studies in Geophysical Flow Computations," Proceedings of the Joint JSME-ASME Fluid Mechanics Meeting, August 14-18, 1995, Hilton Head, SC, Session F137, 1995.

Appendix – Microsoft Excel VBA Macro for Spring-Mass MMS Demonstration

```

Sub SpringM()
'
' --- check that only one area is selected ---
areaCount = Selection.Areas.Count
' --- and EIGHT rows are selected ---
If areaCount <= 1 Then
  RowCount = Selection.Rows.Count
  If RowCount < 8 Then
    MsgBox "Select only EIGHT rows"
    Exit Sub
  ElseIf RowCount > 8 Then
    MsgBox "Select only EIGHT rows"
    Exit Sub
  End If
Else
  MsgBox "Select only one Area"
  Exit Sub
End If
Set col = Selection.Areas(1)
'   MsgBox "xcolumn Address " & xcol.Address
Column = Selection.Areas(1).Column
'   MsgBox "xcolumn Number " & Selection.Areas(1).Column
Row = Selection.Areas(1).Row
'   MsgBox "xcolumn Number " & Selection.Areas(1).Column

  alpha = Cells(Row, Column).Value
  beta = Cells(Row + 1, Column).Value
  Omega = Cells(Row + 2, Column).Value
  TimeEnd = Cells(Row + 3, Column).Value
  nsteps = Cells(Row + 4, Column).Value
  nout = Cells(Row + 5, Column).Value
  Qcase = Cells(Row + 6, Column).Value
  ErrId = Cells(Row + 7, Column).Value
'
  ttime = 0
  deltat = TimeEnd / nsteps
  DtOm2 = (deltat * Omega) ^ 2
'
  x0 = alpha
  FT = QofT(ttime, Qcase, alpha, beta, Omega)
  x1 = (deltat * (deltat * FT + 2 * beta) + (2 - DtOm2) * x0) / 2
  If ErrId = 1 Then
'   *** Sign (Minus) Error ID = 1 ***
    x1 = (deltat * (deltat * FT - 2 * beta) + (2 - DtOm2) * x0) / 2
  End If
'
' --- assign the new column numbers ---
  Xout = Column + 1
  Yout = Column + 2
' --- assing column titles to new columns ---
  Cells(1, Xout) = "Time"
  Cells(1, Yout) = "Numerical"

```

```

' --- write initial values ---
  Cells(2, Xout).Value = ttime
  Cells(2, Yout).Value = x0
  ttime = ttime + deltat
  Cells(3, Xout).Value = ttime
  Cells(3, Yout).Value = x1
'
  kloop = 3
  iout = nsteps / nout
For i = 1 To nsteps Step 1
'  ttime = ttime + deltat
  FT = QofT(ttime, Qcase, alpha, beta, Omega)
  x2 = FT * deltat ^ 2 + (2 - DtOm2) * x1 - x0
  If ErrId = 2 Then
'      *** Number (2) Error ID = 2 ***
    x2 = FT * deltat ^ 2 + (1 - DtOm2) * x1 - x0
  ElseIf ErrId = 3 Then
'      *** wrong algorithm Error ID = 3 ***
    x2 = (FT * deltat ^ 2 + 2 * x1 - x0) / (1 + DtOm2)
  End If
  ttime = ttime + deltat
' --- only output every iout cycles ---
  remain = i Mod iout
  If remain = 0 Then
    kloop = kloop + 1
    Cells(kloop, Xout).Value = ttime
    Cells(kloop, Yout).Value = x2
  End If
  x0 = x1
  x1 = x2
Next i
Exit Sub
End Sub
Function QofT(ttime, Qcase, alpha, beta, Omega)
If Qcase = 0 Then
  QofT = 0
ElseIf Qcase = 1 Then
  QofT = alpha * Omega ^ 2
ElseIf Qcase = 2 Then
  A = alpha - 1
  B = beta / ((alpha - 1) * Omega)
  QofT = (Omega ^ 2) * (1 + A * (B ^ 2 + 1) * Exp(ttime * B * Omega))
ElseIf Qcase = 3 Then
  A = alpha
  B = beta
  QofT = (A + B * ttime) * Omega ^ 2
End If
End Function

```