© Freund Publishing House Ltd. International Journal of Nonlinear Sciences and Numerical Simulation 5(1), 67-78,2004

Interpolation Based Numerical Procedure for Solving Two-Point Nonlinear Boundary Value Problems

D. S. Sophianopoulos

Department of Civil Engineering, School of Engineering, University of Thessaly, Pedion Areos, 383 34 Volos, Greece, Email: dimisof@central.ntua.gr

P. G. Asteris

Faculty of Technological Applications, Technological Educational Institution of Athens, Greece, Email: asteris@teiath.gr

Abstract

The present paper offers a simple, efficient and accurate straightforward numerical method for solving two-point nonlinear boundary value problems (NBVPs), which are frequently encountered in a broad discipline of sciences. The proposed technique is based on interpolation procedures and utilizes the capabilities of readily applicable functions, which are embedded in modern commercial mathematical software *(Mathematica)*. The whole approach is computationally effortless, does not contain the drawbacks of other existing methods related to the solution of NBVPs and under the restriction of a sufficient guess of the interpolation domains is capable of tackling a variety of such problems, originated from different scientific areas. Several applications of the method produce results in excellent agreement with existing ones of the relevant literature, the latter being the product of far more sophisticated algorithms.

Keywords: Boundary value problems, Numerical analysis, Interpolation, Mathematica

1. Introduction

Strongly nonlinear multi-point and especially two-point boundary value problems (NBVPs) very often appear in a variety of problems arising in a broad discipline of sciences, such as mathematics, theoretical physics, solid and fluid mechanics, control and optimization theory etc. In this class of problems, for a set of nonlinear ordinary differential equations some boundary conditions are specified at the initial value of the independent variable and the remaining ones at its terminal value[6,11].

Although there exist many numerical as well as semi-analytical methods for solving two-point NBVPs[3, 8, 13, 16, 20], no precise and widely applicable methods are available to date, that can treat all different types of such problems. Furthermore, neither the existence nor the uniqueness of a solution is always guaranteed, while additional difficulties may appear due to possible singularities involved. Among the methods described in the relevant literature, the most popular ones are the initial value methods (shooting and its variants, power series, double Fourier series), the finite difference methods, various versions of Newton's method including quasi-linearization[13, 16], integral equation methods, function space approximation methods (Galerkin, collocation, generalized Ritz)[8], general iterative methods[17], perturbation methods[5, 10], Padè approximation methods, FE methods coupled with iterative techniques and the recently developed method of locally transversal linearization[14].

In a multitude of cases and for engineering problems in particular, the direct application of one of the aforementioned methods often has as a consequence the distraction from the essence of the physical problem dealt with; this occurs since most of these methods are associated with sophisticated numerical algorithms and require the decomposition of higher order differential equations into a system of 1st order ones. An

efficient way to overcome this obstacle is to resort to modern commercial mathematical software and take advantage of the strong capabilities of its powerful embedded functions, a task that can be achieved without significant computational effort.

In doing this, the present paper offers a simple yet efficient numerical algorithm for the solution of two-point nonlinear boundary value problems using *Mathematica*[1,2] and more specifically utilizes the embedded functions *ListInterpolation* (or *Interpolation*), *NDSolve* and *FindRoot*. As it will be presented in detail below, the proposed approach does not necessitate the rewriting of the higher order ODEs into a system of 1st order ones; moreover, the warning mechanisms of *Mathematica* may detect failure of the method, as far as the interpolation domains are concerned, and also possibly existing singularities. In addition to the above, solving the corresponding linearized problem may also produce lower or upper bound estimates of the solutions sought (for the original nonlinear problem), a fact that can be comprehensively accounted for in the method presented herein. Its validity, accuracy and wide range of applicability is demonstrated in what follows through numerical examples dealing with two NBVP's which posses an exact solution, with the large postbuckling response of a nonconservative continuous system[7, 9, 12, 18], with the equation of Troesch[15, 21] and finally with the Falkner -Skan equation[4, 8].

2. Description of the Method

We consider the most general form of a two-point nonlinear boundary value problem, which can be described by the following nth order differential equation

$$\left.\begin{array}{l} y^{(n)} = f\left(y^{(n-1)}, y^{(n-2)}, ..., y'', y', y, x\right) \\ f : [x_0, x_N] x R^n \to R^n \quad , \quad f \text{ differentiable} \end{array}\right\} \tag{1}$$

associated with n – generally also nonlinear – boundary conditions of the form

$$L_i\left(y^{(n-1)}, y^{(n-2)}, ..., y'', y', y, x\right)\Big|_{x=x_0} = 0 \qquad 1 \le i \le r \tag{2a}$$

$$R_j\left(y^{(n-1)}, y^{(n-2)}, ..., y'', y', y, x\right)\Big|_{x=x_N} = 0 \qquad r+1 \le j \le n \ , \ i+j=n \tag{2b}$$

The *NDSolve* command embedded in *Mathematica*, for the solution of such a higher order DE requires a sufficient set of initial conditions and the range to solve for as inputs. The main concept of the method outlined herein is based on the numerical evaluation of the unknown initial conditions at one of the boundaries. Assuming that this will be performed at the left (lower) boundary x_o , one must determine the values of j initial conditions, which are considered equal to:

$$L_j\left(y^{(n-1)}, y^{(n-2)}, ..., y'', y', y, x\right)\Big|_{x=x_0} = A_j \tag{3}$$

If A_j are thought of as unknowns, one may specify j windows-domains $\left[A_j^{\ell}, A_j^{r}\right]$ in which each A_j is postulated to lie, and then employ *NDSolve* in order to subsequently determine the values of R_j, as in eq.(2b), within the domains chosen above, as functions of A_j incrementally, with a step equal to A_j^s . Thereafter a multidimensional table is created, containing all values of R_j, which have been computed via *NDSolve*. The tabulated results are used to create approximate functions through the *Mathematica* command *ListInterpolation* and the interpolated results are stored as functions of A_j, say $F(A_j)$. The remaining task is to determine those values of A_j that satisfy all the conditions valid at the right (remote) boundary. This is achieved by implementing the *FindRoot* command and seeking solutions for F(A_j)=0 within the previously specified domains.

Evidently, the most difficult task of the whole procedure is the appropriate choice of all these domains, since a bad range for A_j will eventually lead to failure of the numerical scheme. Nevertheless, if one or more of these domains do not contain the correct value of the corresponding A_j, *Mathematica* will produce a warning and afterwards automatically apply *Extrapolation*, which may provide an estimate of the sought A_j values and surely indicate that a bad choice has been made by the user. If again the original nth order D.E. has a singularity, *NDSolve* is capable of detecting it and thus provide useful information

to the user, about the way the algorithm must be adjusted. The general appearance of a *Mathematica* Notebook describing the overall features of the method is listed below:

3. Applications, Numerical Results and Discussion

The simplicity, accuracy, validity and wide range of applicability of the proposed method, which has recently been applied successfully to BVPs related with the large postbuckling response of beams made of nonlinearly elastic material[19], will be demonstrated in this section, by solving five (5) two-point nonlinear BVPs from different scientific areas.

3.1. Two-Point NBVPs with exact solutions

The first application of the method is performed on two (2) BVPs of purely mathematical nature, which posses exact solutions and are given by:

$$\left.\begin{aligned} y''(x) &= y^2(x) + 2\pi^2\cos(2\pi x) - \sin^4(\pi x) \\ 0 &\le x \le 1 \\ y(0) &= y(1) = 0 \end{aligned}\right\} \quad (4a)$$

with an exact solution

$$y(x) = \sin^2(\pi x) \quad (4b)$$

and

$$\left.\begin{aligned} &y''(x) = y^3(x) - y(x)y'(x) \quad , \quad 1 \le x \le 2 \\ &y(1) = \frac{1}{2} \quad , \quad y(2) = \frac{1}{3} \end{aligned}\right\} \quad (5a)$$

with an exact solution

$$y(x) = \frac{1}{1+x} \quad (5b)$$

The proposed technique, if applied on equations (4a) and (5a), produces accurate results, as depicted in Figs.2 and 3, containing the corresponding Notebooks.

3.2. Exact Elastica postbuckling response of a continuous nonconservative system

Extending a previous recent application of the proposed method [19], the two-point NBVP associated with the postbuckling response of a cantilever beam under partial follower loading is dealt with. The geometry and sign convention of the beam is shown in Fig.4, while the force acting at the tip is related with a nonconservativeness parameter α.

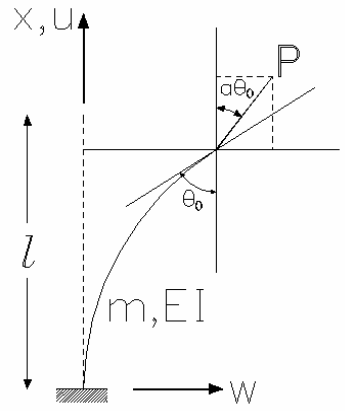

Fig.4 Geometry and sign convention of a cantilever beam under partial follower loading

The differential equation governing the exact large deformation (Elastica) response of the column is given in dimensionless form by the following well known expression, often encountered in the relevant literature[7, 18] :

$$\left.\begin{aligned} &\theta'' + k^2\cos(\alpha\theta_0)\left[1 - \frac{k^2}{\lambda^2}\cos(\alpha\theta_0)\cos\theta\right]\sin\theta \\ &- k^2\sin(\alpha\theta_0)\cos\theta = 0 \ , \ 0 \le x \le 1 \\ &k^2 = \frac{P\ell^2}{EI} \quad , \quad \theta_0 = \theta(1) \end{aligned}\right\} \quad (6)$$

where k is the dimensionless load at the tip, $\theta=\theta(x)$ is the rotation of the tangent at point x of the column axis (considered incompressible), with the axial coordinate referring to the undeformed (perfect) state, while λ is the column's slenderness ratio.

Equation (6) is associated with the following boundary conditions

$$\theta(0) = \theta'(1) = 0 \tag{7}$$

The two-point NBVP defined by eqs.(6) and (7) can be treated with the aid of elliptic integrals, which however are not suitable for establishing postbuckling equilibria; this is the main reason that other approximate techniques have been proposed[7, 9, 18] for the specific problem. Moreover, it has been shown that the starting point of the postbuckling equilibrium paths (i.e. the divergence buckling loads) are the solutions of the following linear stability equation:

$$\cos k = \frac{\alpha}{\alpha - 1} \quad , \quad \alpha < 0.5 \tag{8}$$

while all postbuckling paths posses an asymptote at[7, 18]

$$\theta = \theta cr = \pi / 1 - \alpha \tag{9}$$

The proposed algorithm can directly and reliably deal with the problem, as shown in the corresponding Notebook depicted in Fig. 5.

Through this procedure one may determine all postbuckling equilibrium paths for any desired value of parameter α, as well as compute the deformed configuration of the column for a specific value of α and various tip angles. Typical graphs are presented throughout Figs. 6 and 7 respectively, being in excellent agreement with previous findings[7, 18].

3.3. The equation of Troesch

The problem of Troesch[21], which arose in the investigation of the confinement of a plasma column by radiation pressure, is inherently unstable and although a combination of techniques has been proposed for its solution (multipoint, continuation and perturbation), none of these has been proven sufficiently potent[15]. The two-point NBVP associated with the equation of Troesch is given by:

$$y'' - m^2 \sinh\left(q^2 y\right) = 0 \quad , \quad y(0) = 0 \quad , \quad y(1) = 1 \tag{10}$$

while the corresponding initial value problem

$$u'' = m^2 \sinh\left(q^2 u\right) \quad , \quad u(0) = 0 \, , u'(0) = s \tag{11}$$

has a singularity, approximately located at

$$x_\infty = \frac{1}{qm} log \frac{8m}{qs} + \ldots \tag{12}$$

For instance, if $m = q = \sqrt{5}$ the problem is not singular for $0 \le x \le 1$ if $s \le 0.0539$, since the singularity is located at $x \cong 1.03 > 1$. The corresponding *Mathematica* Notebook, shown in Fig. 8 and containing the solution of the BVP for the above values of *m, q*, yields *y(1)*=1 and $s = 0.0457417 < 0.0539$ at the right boundary, implying accuracy and efficiency.

3.4. Falkner-Skan boundary layer equation

In this last application the two-point NBVP arising in the investigation of laminar boundary layers exhibiting similarity is considered. In particular[4,8], for the case of two-dimensional flows, the equations of the incompressible boundary layer are reduced to:

$$\left.\begin{array}{l} f''' + ff'' + \beta\left[1 - (f')^2\right] = 0 \\ f(0) = f'(0) = 0 \\ \lim\limits_{n_\infty \to \infty} f'(n_\infty) = 1 \end{array}\right\} \tag{13}$$

This is the well-known Falkner-Skan equation, extensively dealt with in the literature, for accelerating ($\beta>0$), constant ($\beta=0$) or decelerating ($\beta<0$) flows ahead from the separation point of zero shear wall. For this specific problem no closed form solutions are known and thus only numerical solutions (mainly shooting) are applicable. The foregoing approach, as shown in the *Mathematica* code and results of Fig. 9, can easily produce reliable solutions for both positive and negative values of β, being in very good agreement with existing ones[4].

4. Conclusions

In this paper a simple and efficient numerical algorithm is presented for the solution of two-point nonlinear boundary value problems. The proposed approach, based on the capabilities of functions embedded in modern commercial mathematical software and mainly on Interpolation procedures, is straightforward, requires minimal computational effort and may be effi-

ciently applied to BVPs originated from a broad spectrum of sciences. Numerical results obtained are found to be in excellent agreement with existing ones, the latter being products of far more sophisticated methods.

References

[1] Abel, M.A., Braselton, J.P. *Differential Equations with Mathematica, 2nd Edition,* Academic Press, New York, 1995.

[2] Bahder, T.B. *Mathematica for Scientists and Engineers,* Addison-Wesley, 1995.

[3] Cash, J.R. A comparison of some global methods for solving two-point boundary value problems, *Appl. Math. Comput.,* **31** (1989): 449 – 462.

[4] Cebeci, T. Keller, H.B. Shooting and Parallel Shooting Methods for solving the Falkner-Skan Boundary Layer Equation, *Journal of Computational Physics,* **7**(2) (1971).

[5] Cole, J.D. *Perturbation Methods in Applied Mathematics,* Blaisdell, Waltham, MA, 1968.

[6] Ha, S.N. A Nonlinear Shooting Method for Two-Point Boundary Value Problems, *Computers and Mathematics with Applications,* **42**(10-11) (2001): 1411 – 1420.

[7] Kandakis, G., Kounadis, A.N. On the large postbuckling response of nonlinear continuous systems, *Archive of Applied Mechanics,* **62** (1992): 256 – 265.

[8] Keller, H.B. *Numerical Methods for Two-Point Boundary Value Problems,* Dover Publications Inc., New York, 1992.

[9] Kounadis, A.N. An efficient and simple approximate technique for solving nonlinear initial and boundary-value problems, *Computational Mechanics,* **9** (1992): 221 – 231.

[10] Ling, R. Singular Perturbation for Non-Linear Boundary Value Problems, *Int. J. Math & Math. Sci.,* **2** (1979): 61 – 68.

[11] Mazzia, F., Sgura, I. Numerical Approximations of Nonlinear BVPs by means of BVMs, *Applied Numerical Mathematics,* **42**(1-3) (2002): 337 –352.

[12] Plaut, R.H. Post-Buckling Behavior of Continuous Nonconservative Elastic Systems, *Acta Mechanica,* **30** (1978): 51 – 64.

[13] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. *Numerical Recipes,* Cambridge University Press, New York, 1986.

[14] Ramachandra, L.S., Roy, F. A New Method for Nonlinear Two-Point Boundary Value Problems in Solid Mechanics, *Journal of Applied Mechanics (ASME),* **68** (2001): 776 – 786.

[15] Roberts, S.M., Shipman, J.S. Solution of Troesch's two-point boundary value problem by a combination of techniques, *Journal of Computational Physics*, **10**(2) (1972): 232 – 241.

[16] Ronto, M., Samoilenko, A.M. *Numerical – Analytical Methods in the Theory of Boundary Value Problems,* World Scientific Publishing Co., 1997.

[17] Sharidharan, R., Agarwal, R.P. General Iterative Methods for Nonlinear Boundary Value Problems, *Journal of the Australian Mathematical Society,* Ser.B **37** (1995): 58 – 85.

[18] Sophianopoulos, D.S., Kokkinis, V. Static and dynamic stability of Beck's column under partial follower loading, *4th European Conference on Structural Dynamics, EURODYN '99*, Czech Republic, 1999.

[19] Sophianopoulos, D.S., Konstantakopoulos, T.G. A Root-Finding Approach for the Solution of Two-Pont Nonlinear Boundary Value Problems Arising in Engineering Applications, *5th World Congress on Computational Mechanics, WCCM V*, Austria, 2002.

[20] Stoer, J., Burlisch, R. *Introduction to Numerical Analysis,* Springer-Verlag, New York, 1980.

[21] Troesh, B.A. Intrinsic difficulties in the numerical solution of a boundary value problem, *Space Tech. Labs, Tech. Note NN-142*, 1960.

(Clearing the values of all parameters used in the notebook *)*

ClearAll[x_0, x_N, f, y, x, y_1, y_2, …, y_n, P_0, P_1, P_2, …, P_n, sol, i, j, L_1, L_2,…, L_i, R_{i+1}, R_{i+2}, …, R_j, A_1, A_2, …, A_j, A_1^ℓ, A_2^ℓ, …, A_j^ℓ, A_1^r, A_2^r, …, A_j^r, A_1^s, A_2^s, …, A_j^s, F, …]

*(*Input of problem's specific external parameters – known values*)*

……

(Definition of all derivatives of function y *)*

$y_1 = \partial_x\ y[x]$

$y_2 = \partial_x\ y_1$

……

$y_n = \partial_x\ y_{n-1}$

(Original Differential Equation *)*

f = …..

(Known boundary conditions at the "left" – functions of x, y, y_1, y_2, …, y_n at $x=x_0$ *)*

L_1 = …

L_2 = …

……

L_i = …

(Implementation of NDSolve *)*

sol = NDSolve[{f== 0, $L_1 == 0$, $L_2 == 0$, … , $L_i == 0$, $L_{i+1} == A_1$, $L_{i+2} == A_2$, … , $L_n == A_j$}, y, {x, x_0, x_N}]

(Computation of the values of y and its derivatives at the "right" boundary *)*

P_0 = y[x_N]/sol.

P_1 = y_1[x_N]/sol.

.

.

.

Pn-1=y_{n-1}[x_N]/sol.

(Computation of boundary conditions at the "right" using the values of P_0, P_1, …, P_{n-1} as functions of A_1, A_2, …, A_j *)*

$R_{i+1}[A_1\_, A_2\_, …, A_j\_]$ = …

$R_{i+2}[A_1\_, A_2\_, …, Aj\_]$ = …

……

……

$R_j[A_1\_, A_2\_, …, A_j\_]$ = …

(Creation of multidimensional table *)*

Table[{A_1, A_2, … , A_j, $R_{i+1}[A_1, A_2, … , A_j]$, $R_{i+2}[A_1, A_2, … , A_j]$, … , $R_j[A_1, A_2, … , A_j]$}, {A_1, A^l_1, A^r_1, A^s_1}, {A_2, A^l_2, A^r_2, A^s_2}, … , {A_j, A^l_j, A^r_j, A^s_j}]

(Interpolation of tabulated results and setting as a function of A_1, A_2, …, A_j *)*

F = ListInterpolation[%]

$F[A_1, A_2, …, A_j]$

(Implementation of FindRoot for determining the correct values of A_1, A_2, …, A_j *)*

FindRoot[$F[A_1, A_2, …, A_j]$==0,{A_1, A_2, …, A_j,{A_1^ℓ, A_1^r}, {A_2^ℓ, A_2^r}, …, {A_j^ℓ, A_j^r}}]

Fig.1. Mathematica Notebook describing the general principles of the proposed method

```
ClearAll[f, x, A, Al, Ar, As, F1, F2, equ, ww, a1, a2]
equ = f''[x] - f[x]^2 - 2 π^2 Cos[2 π x] + Sin[π x]^4;
Al = -0.01;
Ar = 0.01;
As = N[(Ar - Al)/10000];
F1[A_] := f[1] /. NDSolve[{equ == 0, f[0] == 0, f'[0] == A}, f, {x, 0, 1}];
Table[{A, F1[A]}, {A, Al, Ar, As}];
F2 = Interpolation[%];
F2[A];
FindRoot[F2[A] == 0, {A, {Al, Ar}}];
ww = NDSolve[{equ == 0, f[0] == 0, f'[0] == A /. %}, f, {x, 0, 1}];
s = f[1] /. ww
a1 = Plot[Evaluate[f[x] /. ww], {x, 0, 1}, PlotLabel → Function f[x],
  AxesLabel → {"x", "f(x)"},
  TextStyle → {FontSlant → "Italic", FontSize → 16, FontFamily → "Times"}]
a2 = Plot[Sin[π x]^2, {x, 0, 1}, AxesLabel → {"x", "Sin[π x]^2"},
  TextStyle → {FontSlant → "Italic", FontSize → 16, FontFamily → "Times"}]
Show[a1, a2]
{4.3695967232165784 *^-14}  ——————  f(1) ≅ 0
```

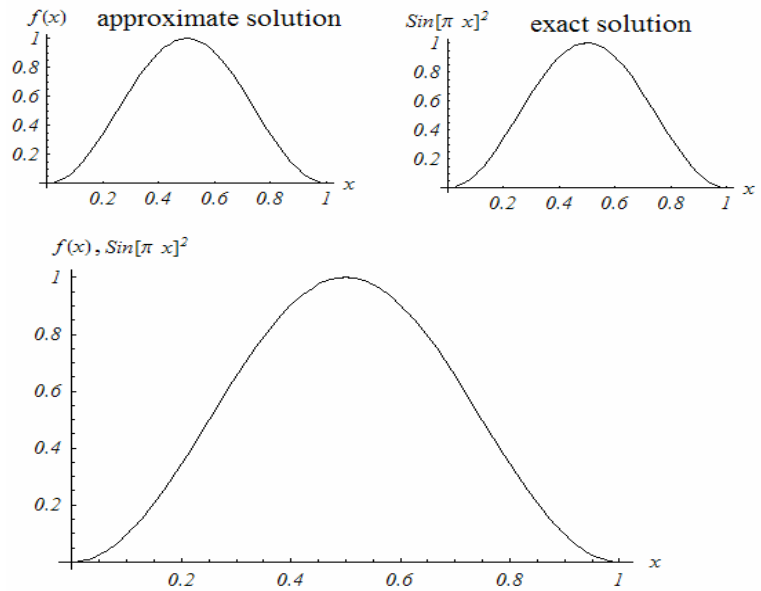

Fig.2 Code and results corresponding to a BVP with an exact solution (1st example)

```
ClearAll[f, x, A, Al, Ar, As, F1, F2, equ, ww, a1, a2]
equ = f''[x] - f[x]^3 + f[x] f'[x];
Al = -0.3;
Ar = -0.2;
As = N[(Ar - Al)/10000];
F1[A_] := f[2] /. NDSolve[{equ == 0, f[1] == 0.5, f'[1] == A}, f, {x, 1, 2}];
Table[{A, F1[A]}, {A, Al, Ar, As}];
F2 = Interpolation[%];
F2[A];
FindRoot[F2[A] == 1/3, {A, {Al, Ar}}];
ww = NDSolve[{equ == 0, f[1] == 0.5, f'[1] == A /. %}, f, {x, 1, 2}];
s = f[2] /. ww
a1 = Plot[Evaluate[f[x] /. ww], {x, 1, 2}, PlotLabel → approximate solution,
    AxesLabel → {"x", "f(x)"},
    TextStyle → {FontSlant → "Italic", FontSize → 16, FontFamily → "Times"}]
a2 = Plot[1/(1 + x), {x, 1, 2}, AxesLabel → {"x", "1/(1 + x)"}, PlotLabel → exact solution,
    TextStyle → {FontSlant → "Italic", FontSize → 16, FontFamily → "Times"}]
Show[a1, a2]
```

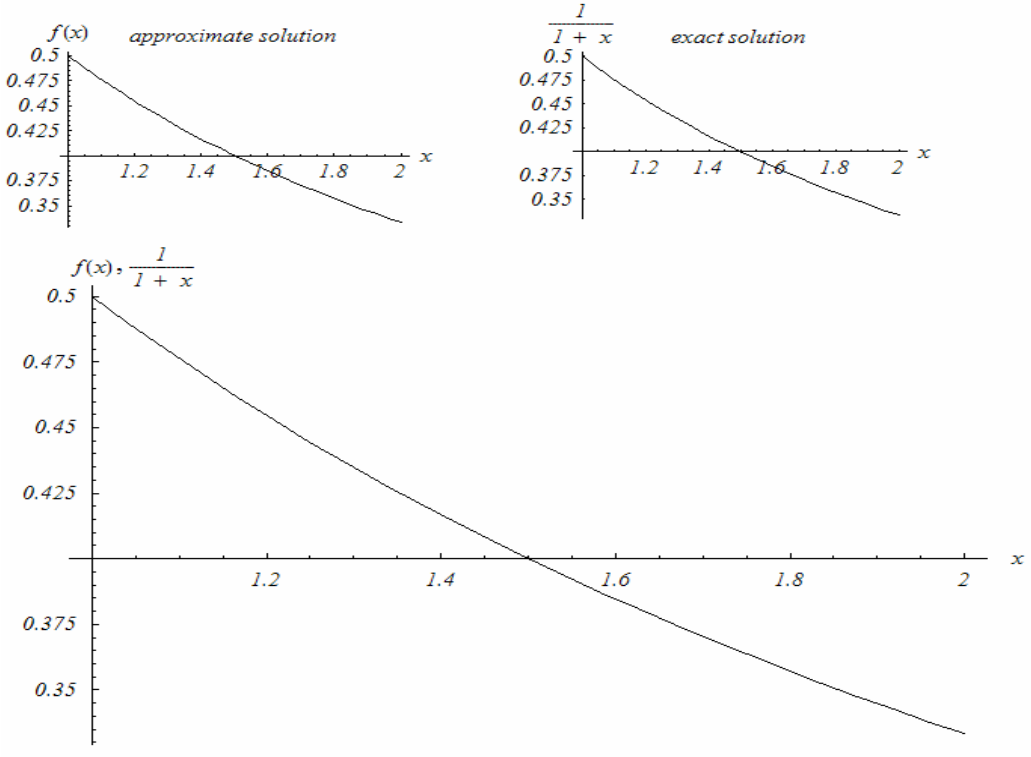

Fig.3 Code and results corresponding to a BVP with an exact solution (2nd example)

```
ClearAll[Θ, θo, k, λ, equat, equat2, α, As, Al, Ar, F1, F2, A, x, rr, ss, θcr, ko, kn, kk, aa]
λ = 30;
α = 0.20;
θcr = π / (1 − α)
ko = ArcCos[ α / (α − 1) ]
θo = 0.05;
kn = 1.2 ∗ko;
kk = (kn − ko) / 100;
equat = −cos(Θ(x)) sin(α θo) k^2 + cos(α θo) (1 − k^2 cos(α θo) cos(Θ(x)) / λ^2) sin(Θ(x)) k^2 + Θ''(x);
Al = 0.001;
Ar = 0.5;
As = (Ar − Al) / 100 // N ;
F1[A_, k_] := Θ[1] /. NDSolve[{equat == 0, Θ[0] == 0, Θ'[0] == A}, Θ, {x, 0, 1}]
Table[{A, k, F1[A, k]}, {A, Al, Ar, As}, {k, ko, kn, kk}];
F2 = ListInterpolation[%];
F2[A, k];
ff = FindRoot[{F2[A, k] == θo, F2[A, k] == θo}, {A, {Al, Ar}}, {k, {ko, kn}}]
aa = A /. ff ;
kk = k /. ff ;
equat2 = −cos(Θ(x)) sin(α θo) kk^2 + cos(α θo) (1 − kk^2 cos(α θo) cos(Θ(x)) / λ^2) sin(Θ(x)) kk^2 + Θ''(x);
rr = NDSolve[{equat2 == 0, Θ[0] == 0, Θ'[0] == aa}, Θ, {x, 0, 1}];
ss = Θ[1] /. rr
Plot[Evaluate[Θ[x] /. rr], {x, 0, 1}, AxesLabel → {"x", "θ(x)"}]
```

asymptote of postbuckling equilibrium path

nonconservativeness parameter

divergence buckling load

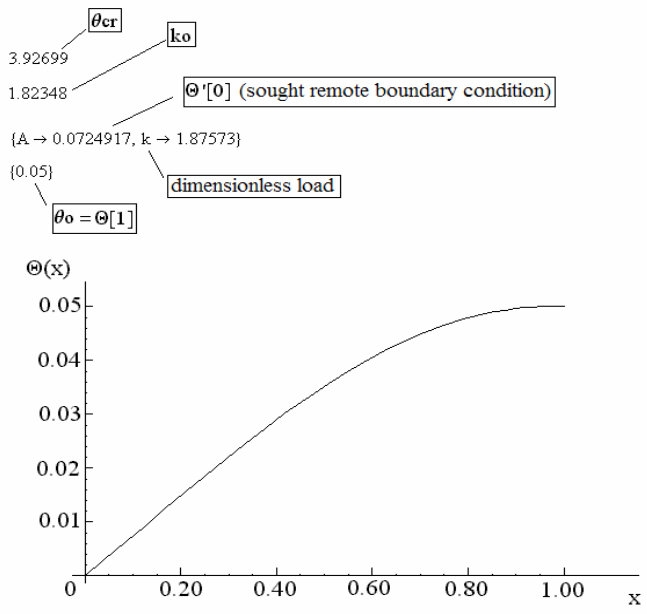

Fig.5 Mathematica Notebook for the NBVP corresponding to the postbuckling behavior of the beam depicted in Fig. 4

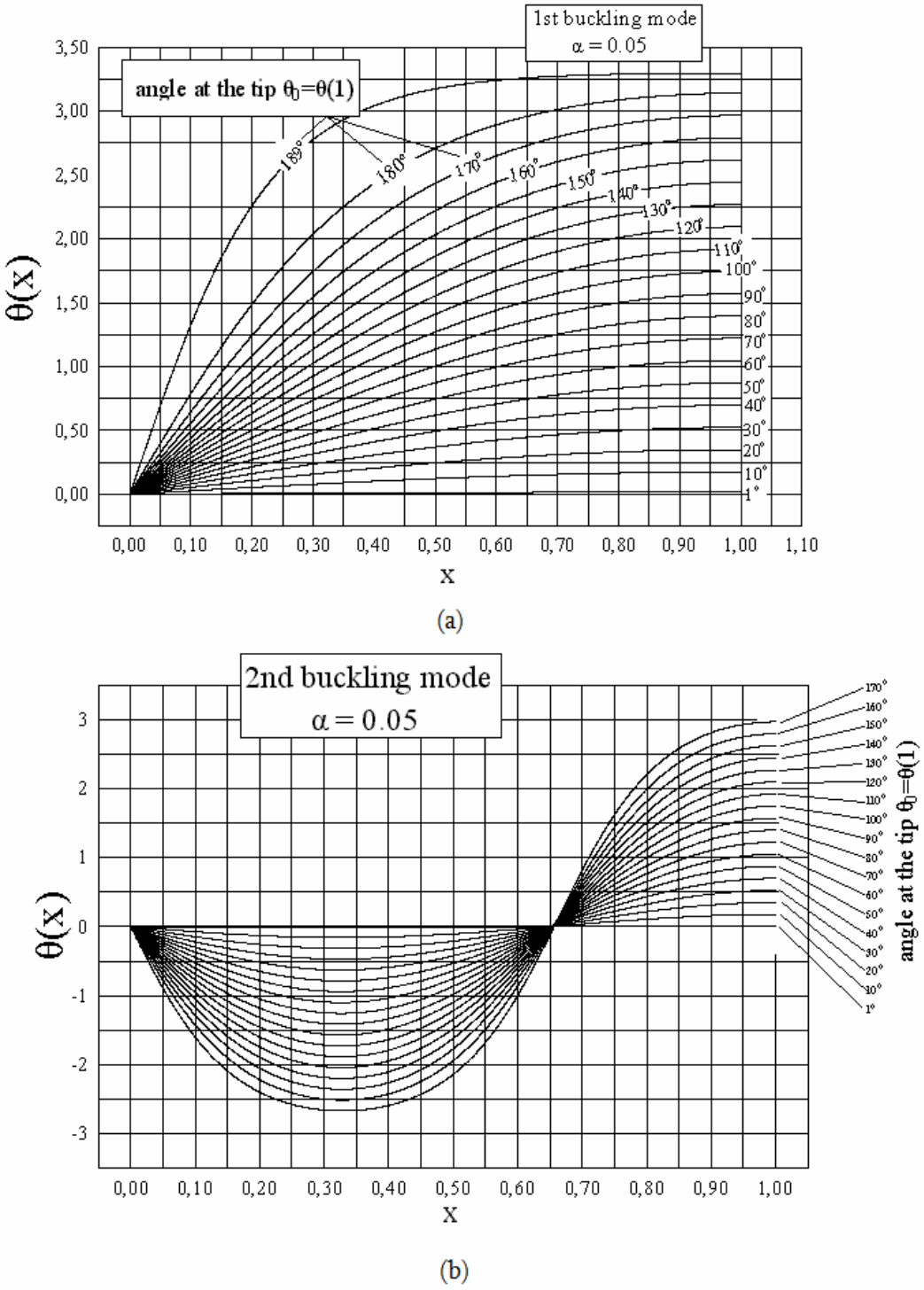

Fig.7 First (a) and second (b) buckling mode of the beam shown in Fig. 4, for $\alpha = 0.05$ and various values of the angle at the tip

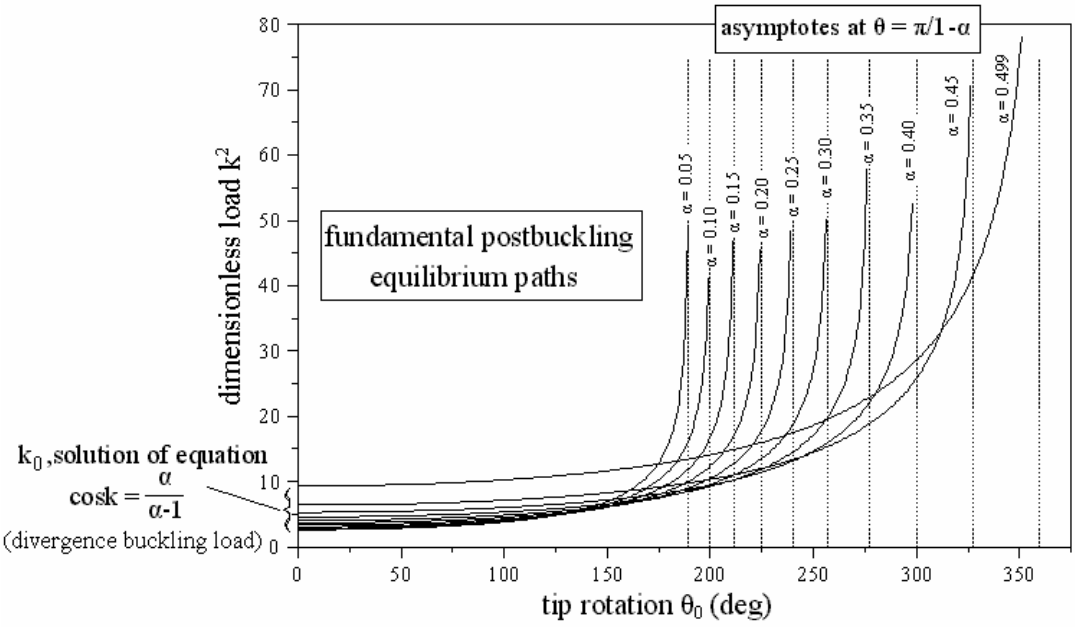

Fig.6 Fundamental postbuckling equilibria of the beam shown in Fig. 4, for various values of noncon-servativeness parameter α

```
ClearAll[f, x, A, Al, Ar, As, F1, F2, equ, ww]
equ = f''[x] - 5 * Sinh[5 * f[x]]
Al = 0.01;
Ar = 0.05;
As = (Ar - Al) / 10000 // N;
F1[A_] := f[1] /. NDSolve[{equ == 0, f[0] == 0, f'[0] == A}, f, {x, 0, 1}];
Table[{A, F1[A]}, {A, Al, Ar, As}];
F2 = Interpolation[%];
F2[A];
FindRoot[F2[A] == 1, {A, {Al, Ar}}]
ww = NDSolve[{equ == 0, f[0] == 0, f'[0] == A /. %}, f, {x, 0, 1}];
s = f(1) /. ww
Plot[Evaluate[f[x] /. ww], {x, 0, 1}, PlotLabel → Function f(x), AxesLabel → {"x", "f(x)"},
  TextStyle → {FontSlant → "Italic", FontSize → 16, FontFamily → "Times"}]
```

```
{A → 0.0457417} ——— f'[0]
{1.} ——————————— f[1]
```

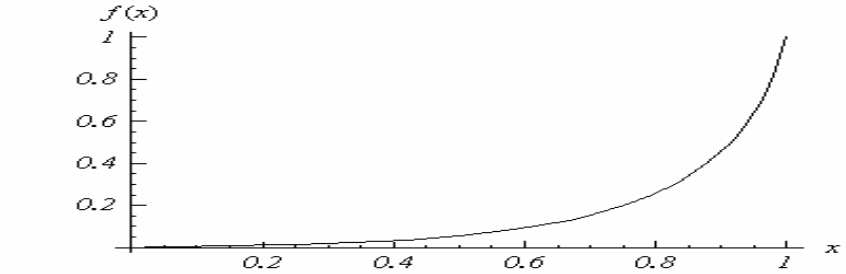

Fig.8 Mathematica Notebook for Troesch's equation ($m = q = \sqrt{5}$)

```
ClearAll[n, f, β, A, Al, Ar, As, W1, W2, equ]
β = −0.19;
n = 10;
equ = f'''[x] + f[x] * f''[x] + β * (1 − f'[x]^2);
Al = 0;
Ar = 0.5;
As = (Ar − Al) / 1000 // N;
W1[A_] := f'[n] /. NDSolve[{equ == 0, f[0] == 0, f'[0] == 0, f''[0] == A}, f, {x, 0, n}];
Table[{A, W1[A]}, {A, Al, Ar, As}];
W2 = Interpolation[%];
W2[A];
FindRoot[W2[A] == 1, {A, {Al, Ar}}]
NDSolve[{equ == 0, f[0] == 0, f'[0] == 0, f''[0] == A /. %}, f, {x, 0, n}];
pl1 = Plot[Evaluate[f'[x] /. %], {x, 0, n}, PlotRange → {0, 1.2}];
```

```
ClearAll[n, f, β, A, Al, Ar, As, W1, W2, equ]
β = 1;
n = 10;
equ = f'''[x] + f[x] * f''[x] + β * (1 − f'[x]^2);
Al = 1.2;
Ar = 1.4;
As = (Ar − Al) / 1000 // N;
W1[A_] := f'[n] /. NDSolve[{equ == 0, f[0] == 0, f'[0] == 0, f''[0] == A}, f, {x, 0, n}];
Table[{A, W1[A]}, {A, Al, Ar, As}];
W2 = Interpolation[%];
W2[A];
FindRoot[W2[A] == 1, {A, {Al, Ar}}]
NDSolve[{equ == 0, f[0] == 0, f'[0] == 0, f''[0] == A /. %}, f, {x, 0, n}];
pl2 = Plot[Evaluate[f'[x] /. %], {x, 0, n}, PlotRange → {0, 1.2}];
```

```
Show[pl1, pl2, TextStyle → {FontSlant → "Italic", FontSize → 16, FontFamily → "Times"}]
```

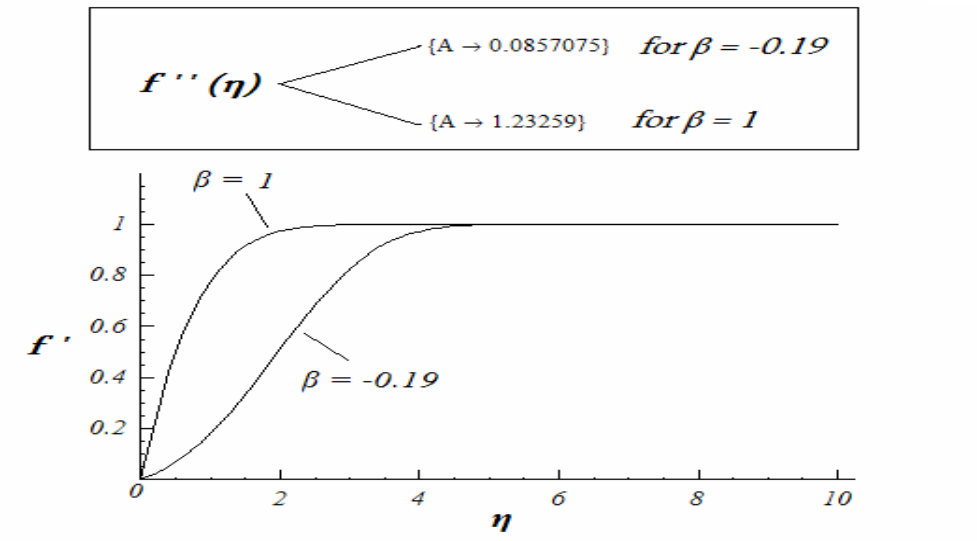

Fig.9 Mathematica Notebook for the Falkner-Skan equation and two values of parameter β