# A Robust Numerical Method for a Singularly Perturbed Nonlinear Initial Value Problem

Jacob W. Adkins
jadkin31@kent.edu
**Honors Thesis**
Advisor: Dr. Relja Vulanovic
Kent State University at Stark, Ohio, USA

November 16, 2017

## 1 Introduction

Ordinary Differential Equations (ODEs) are used as mathematical models for many different phenomena in various fields of study. Civil engineers use them to describe the bending and reflection of beams. Nuclear engineers apply them when studying diffusion of neutrons in an atomic pile to produce nuclear energy [1].

Suppose we want to know how a quantity of some medium changes in relation to time (for example, the concentration of carbon dioxide in an enclosed space). This type of problem can be solved using an ODE, which allows us to predict the quantity in the future, knowing its value now and knowing that its rate of change depends on the current quantity. The present value of the quantity is also called an *initial value* and this is why the whole problem is an Initial Value Problem (IVP). An IVP consists of a differential equation and an initial value (the value of the quantity at the beginning of our observations). IVPs are commonly introduced in relation to simple harmonic motion or oscillation in physics. Moreover, the well-known Laplace's Equation has applications in electricity, potential theory, gravitation, and aerodynamics - extending to rocket flight [2].

If the differential equation contains a small positive parameter, $\varepsilon \ll 1$, multiplying the highest derivative in the equation, it is known as a singularly perturbed differential equation. In this thesis, I consider a singularly perturbed IVP,

$$\varepsilon u' + a(x,u) = 0, \quad x \in (0,1), \quad u(0) = u_0, \tag{1}$$

where $a$ is a sufficiently smooth function satisfying

$$a_u(x,u) \geq a_* > 0, \quad x \in [0,1], \quad u \in \mathbb{R}. \tag{2}$$

Singularly perturbed IVPs have applications in the study of vibrations, chemical reactions, and electrical circuits [3].

The problem in (1) is to find a $C^1[0,1]$-solution $u$ which satisfies both the given ODE and initial value. Conventionally, there does not exist an analytic method to solve this problem due to the generality of $a(x,u)$ [2]. In fact, only for specific functions will a deterministic algorithm exist. For this reason, a numerical method is required [3]. Moreover, solutions of singularly perturbed differential equations change abruptly on narrow intervals called *layers*. Because of layers, standard numerical procedures are ineffective and special methods need to be used. The crucial result of the thesis is the construction of one such special numerical method, namely, the backward Euler scheme [4] on a modified Shishkin Mesh [5]. This will improve the result from [4] along the lines of [5].

## 2 Analysis of the Continuous Solution

We show that (1) satisfies the Maximum Principle [6] by first examining the linear case. Consider the linear differential operator

$$L[u] = \varepsilon u' + \tilde{a}(x)u.$$

**Definition 2.1** *We say that L satisfies the maximum principle if* $L[u] \geq 0$ on $[0,1]$ *and* $u(0) \geq 0$ *imply that* $u \geq 0$ on $[0,1]$.

**Lemma 2.2** *If* $\tilde{a}(x) \geq 0$ for $x \in [0,1]$*, then L satisfies the maximum principle.*

*Proof: For* $u \in C^1[0,1]$*, assume* $L[u] \geq 0$ on $(0,1)$ and $u(0) \geq 0$*, but there exists a point* $x_* \in [0,1]$ *such that* $u(x_*) < 0$*. This implies there exists a point* $\tilde{x}$ *such that* $u'(\tilde{x}) < 0$*, and* $u(\tilde{x}) < 0$*. See Figure 1. Then*

$$L[u](\tilde{x}) = \varepsilon u'(\tilde{x}) + \tilde{a}(\tilde{x})u(\tilde{x}) < 0,$$

*which is a contradiction, and L satisfies the Maximum Principle.*

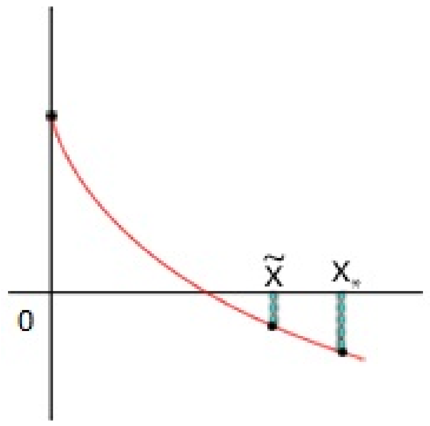

Figure 1: Existence of $x_*$ implies existence of $\tilde{x}$.

Consider now the non-linear differential operator $T[u] = \varepsilon u' + a(x,u)$.

**Definition 2.3** *We say that $T$ satisfies the maximum principle if*

$$T[u_1] \geq T[u_2] \text{ on } [0,1] \text{ and } u_1(0) \geq u_2(0) \tag{3}$$

*imply $u_1 \geq u_2$ on $[0,1]$.*

**Lemma 2.4** *If $a_u(x,u) \geq 0$ for $x \in [0,1]$, and $u \in \mathbb{R}$ then $T$ satisfies the maximum principle.*

*Proof: Consider $T[u_1] - T[u_2]$. We have*

$$T[u_1]-T[u_2] = \varepsilon u_1' + a(x,u_1) - \varepsilon u_2' - a(x,u_2) = \varepsilon(u_1-u_2)' + a(x,u_1) - a(x,u_2).$$

*Since*

$$a(x,u_1) - a(x,u_2) = \left(\int_0^1 a_u(x, su_1(x) + (1-s)u_2(x))ds\right)(u_1 - u_2)$$

*by the integral form of the Mean Value Theorem, we define*

$$y = u_1 - u_2 \text{ and } L[y] = \varepsilon y' + \left(\int_0^1 a_u(x, su_1(x) + (1-s)u_2(x))ds\right) y.$$

*Because of Lemma 2.2 we have that $L$ satisfies the maximum principle and, moreover, $L[y] = T[u_1] - T[u_2]$. Therefore if we assume (3), we have*

$$L[y] \geq 0, \text{ and } y(0) \geq 0,$$

*which implies $y \geq 0$, that is, $u_1 \geq u_2$.*

**Lemma 2.5** *Assume (2). Then there exists a constant $C_1$ such that $C_1 \geq u(x)$, $x \in [0,1]$.*

*Proof: By Lemma 2.2,*

$$T[C_1] = \varepsilon C_1' + a(x,C_1) = a(x,C_1).$$

*Assume $C_1 > 0$. Now there exists a point $\theta \in (0, C_1)$* such that

$$a(x,C_1) = a(x,0) + (C_1 - 0)a_u(x,\theta) \geq \alpha + C_1 a_*,$$

*where $\alpha = min_{x\in[0,1]} a(x,0)$. Choosing $C_1$ sufficiently large, such that*

$$T[C_1] \geq \alpha + C_1 a_* \geq T[u] = 0, \text{ and } C_1 \geq u(0),$$

*we have $C_1 \geq u(x)$ by Lemma 2.4.*

**Lemma 2.6** *Assume (2). Then there exists a constant $C_2$ such that $C_2 \leq u(x)$, $x \in [0,1]$.*

*Proof: By Lemma 2.2,*

$$T[C_2] = \varepsilon C_2' + a(x,C_2) = a(x,C_2).$$

*Assume $C_2 \leq 0$. Now there exists a point $\theta \in (0,1)$* such that

$$a(x, C_2) = a(x,0) + (C_2 - 0)a_u(x,\theta) \leq \beta + C_2 a_*,$$

*where $\beta = max_{x\in[0,1]} a(x,0)$. Choosing $C_2$ sufficiently small, such that*

$$T[C_2] \leq \beta + C_2 a_* \leq T[u] = 0, \text{ and } C_2 \leq u(0),$$

*we have $C_2 \leq u(x)$ by Lemma 2.4.*

**Lemma 2.7** *Assume (2). There exists a $C^*$ such that $|u(x)| \leq C^*$, $x \in [0.1]$.*

*Proof: $C^* := \max\{-C_2, C_1\}$ with $C_1$ and $C_2$ like in Lemmas 2.5 and 2.6, respectively.*

From now on, let $C$ denote a generic positive constant which is independent of $\varepsilon$. Let also

$$A(x) := \int_0^x a_u(t, u(t))\, dt.$$

**Lemma 2.8** *The solution $u$ of the initial-value problem* (2) *satisfies*

$$|u^{(i)}(x)| \leq C\left[1 + \varepsilon^{-i} e^{-A(x)/\varepsilon}\right], \quad i = 1,2, \quad x \in [0,1]. \tag{4}$$

*Proof: We have from* (1) *that $u' = -\varepsilon^{-1}a(x,u)$ on* $[0,1]$. *Then because of Lemma 2.7, it holds true that $|u'(x)| \leq C\varepsilon^{-1}$, $x \in [0,1]$. Also, on differentiating* (1)*, we get*

$$-\varepsilon u'' - a_u(x,u)u' = a_x(x,u). \tag{5}$$

*When* (5) *is solved for $u''$, it follows that $|u''(x)| \leq C\varepsilon^{-2}$, $x \in [0,1]$. In particular, we have*

$$|u^{(i)}(0)| \leq C\varepsilon^{-i}, \quad i = 1,2. \tag{6}$$

*Since $a_u(x,u) \geq a_* > 0$, the equation* (5) *has the same structure as the second-order problem considered in [7]. Because of this fact and the estimates in* (6)*, the proof of* (4) *follows like in the proof of Theorem 1 in [7].*

Since $A(x)$ depends on the unknown solution $u$, the estimates in (4) cannot be used in practice. Instead, the following can be derived from (4):

$$|u^{(i)}(x)| \leq C(1 + \varepsilon^{-i}e^{-a_* x/\varepsilon}), i = 1,2, \qquad x \in [0,1]. \tag{7}$$

We now define

$$\tilde{\omega} = \min_{x\in[0,1]} a_u(0, u(x)) \tag{8}$$

and prove the following lemma, which is a new result in this thesis. The proof technique is based on [5].

**Lemma 2.9** *The solution $u$ of the initial-value problem* (1) *satisfies*

$$|u^{(i)}(x)| \le C\left[1 + \varepsilon^{-i} e^{-\tilde{\omega} x/\varepsilon}\right], \quad i = 1, 2, \quad x \in [0, 1]. \tag{9}$$

*Proof: Since in $[0,1]$ there exists a point $\xi$ such that*

$$a_u(x, u(x)) = a_u(0, u(x)) + x a_{xu}(\xi, u(x))$$

*we have,*

$$a_u(x, u(x)) \ge \tilde{\omega} - \omega_1 x, \quad x \in [0, 1], \tag{10}$$

*where $\omega_1$ is a non-negative constant satisfying*

$$a_{xu}(\xi, u(x)) \ge -\omega_1, \quad x \in [0, 1].$$

*From (10) we get the following lower bound on $A$:*

$$A(x) \ge \tilde{\omega} x - \frac{1}{2}\omega_1 x^2, \quad x \in [0, 1],$$

*which implies that*

$$A(x) \ge \tilde{\omega} x - \frac{1}{2}\omega_1 \varepsilon, \quad x \in [0, \sqrt{\varepsilon}].$$

*This combined with (4) gives*

$$|u^{(i)}(x)| \le C(1 + \varepsilon^{-i} e^{-\tilde{\omega} x/\varepsilon + \omega_1/2}), \quad i = 1, 2, \qquad x \in [0, \sqrt{\varepsilon}].$$

*It follows from here that*

$$|u^{(i)}(x)| \le C(1 + \varepsilon^{-i} e^{-\tilde{\omega} x/\varepsilon}), \quad i = 1, 2, \qquad x \in [0, \sqrt{\varepsilon}]. \tag{11}$$

*When $x \in [\sqrt{\varepsilon}, 1]$ we get from (7) that*

$$|u^{(i)}(x)| \le C(1 + \varepsilon^{-i} e^{-a_* x/\varepsilon}) \le C, \quad i = 1, 2.$$

*Thus,*

$$|u^{(i)}(x)| \le C \le C(1 + \varepsilon^{-i} e^{-\tilde{\omega} x/\varepsilon}), \quad i = 1, 2, \qquad x \in [\sqrt{\varepsilon}, 1].$$

*Combining this with (11), we obtain (9).*

Note that, $\tilde{\omega}$ is defined in (6) in terms of a solution we do not know and as such, it is not known in general; therefore, it would be more practical to use (9) with $\omega$ replacing $\tilde{\omega}$, where

$$\omega := \min_{|y| \le C^*} a_u(0, y). \tag{12}$$

However $\omega \le \tilde{\omega} \Rightarrow -\tilde{\omega} \le -\omega$, which leads to the following theorem.

**Theorem 2.10** *The solution $u$ of the problem (1) satisfies the following estimates:*

$$|u^{(i)}(x)| \le C(1 + \varepsilon^{-i} e^{-\omega x/\varepsilon}), \quad i = 1, 2, \qquad x \in [0, 1]. \tag{13}$$

The estimates in (7) and (13) show that $u$ has a layer of order $\mathcal{O}(-\varepsilon \ln \varepsilon)$ at $x = 0$. In the layer, the derivatives of $u$ grow without bounds as $\varepsilon$ decreases. Since $\omega \ge a_*$, the estimates in (13) may be sharper than in (7). This will enable an improvement of the Shishkin mesh, resulting in a more accurate numerical solution.

## 3 Preliminary Numerical Experiments

Traditional numerical methods such as the Euler methods employ some step size used for iteration through the domain of the function. Here we use step size $h = \frac{1}{n}$ of a uniform mesh. Let $x_i = h * i, i = 0, 1, ..., n$. Employing the Forward Euler Method for (1), we get

$$\varepsilon \frac{y_{i+1} - y_i}{h} + a(x_i, y_i) = 0, \qquad i = 0, 1, ..., n-1, \qquad y_0 = u_0$$

Solving for $y_{i+1}$,

$$y_{i+1} = y_i - \frac{h}{\varepsilon} a(x_i, y_i).$$

Similarly, for the Backward Euler Method, (1) yields,

$$\varepsilon \frac{y_{i+1} - y_i}{h} + a(x_{i+1}, y_{i+1}) = 0, \qquad i = 0, 1, ..., n-1, \qquad y_0 = u_0.$$

If linear, we have $a(x, u) = \tilde{a}(x)u + \tilde{b}(x)$. Then solving for $y_{i+1}$ we get,

$$y_{i+1} = \frac{\varepsilon y_i - h\tilde{b}(x_{i+1})}{\varepsilon + h\tilde{a}(x_{i+1})}. \tag{14}$$

The above Forward and Backward Euler methods on a uniform mesh are traditional numerical methods for solving initial-value problems. We now illustrate that they are non-uniform in $\varepsilon$ by applying them to the problem

$$\varepsilon u' + u + \tilde{b}(x) = 0, \qquad x \in (0, 1), \qquad u(0) = 10, \tag{15}$$

with solution

$$u = 10(1 + e^{-x/\varepsilon}) - (10 + x)e^{-x}, \tag{16}$$

which determines $\tilde{b}(x)$ as

$$\tilde{b}(x) = [10 + x - \varepsilon(9 + x)]e^{-x} - 10.$$

The results are given in Table 1 which shows the maximum absolute errors at the mesh points, $x_i$, calculated using the exact solution.

Table 1: Linear Forward Euler Results

| $\varepsilon$ | h = .1 | .01 | .001 |
|---|---|---|---|
| 1 | .03045 | .00289 | .00028 |
| .1 | 3.63993 | .18953 | .01823 |
| .01 | 3.48541 x $10^{10}$ | 3.678395 | .19198 |
| $10^{-3}$ | 9.04346 x $10^{20}$ | 2.65612 x $10^{96}$ | 3.67879 |

This shows the Forward Euler method cannot yield satisfactory results in $\varepsilon$ because they are not stable uniformly in $\varepsilon$, [3]. Additionally, the Backward Euler method, while stable in $\varepsilon$, has non-uniform errors on an equidistant mesh; see Table 2. Thus, a special discretization mesh, like Shishkin's, is needed [4].

Table 2: Linear Backward Euler Results

| $\varepsilon$ | h = .1 | .01 | .001 |
|---|---|---|---|
| 1 | .02725 | .00286 | .00028 |
| .1 | 1.30233 | .17430 | .01807 |
| .01 | .90520 | 1.32100 | .17661 |
| $10^{-3}$ | .09863 | .90860 | 1.32120 |
| $10^{-4}$ | .00995 | .09900 | .90863 |
| $10^{-5}$ | .00099 | .00998 | .09900 |
| $10^{-6}$ | .00009 | .00099 | .00999 |

## 4 A Robust Numerical Method

To stabilize the results, and address the uniformity issues which arise when the Backward Euler formula to (15), we apply the Shishkin Mesh. The Shishkin Mesh is composed of two uniform meshes, a fine mesh in the layer between zero and the transition point, $\tau$, and a crude mesh outside of the layer. We determine the step size in the fine mesh as $\tau/j$, and as $H = \frac{1-\tau}{n-j}$ in the crude mesh. $\tau$ is determined by

$$\tau = \frac{1}{w}\ \varepsilon\ \ln(n),$$

where $w$ is either $a_*$, based on (7), or $\omega$, based on (13). We assume that $\tau < \frac{1}{2}$ since otherwise, $n$ is unreasonably large. Refer to (14): we use

$$h = \frac{\tau}{j} \qquad \text{for} \qquad i = 0, 1, ..., j$$

and

$$h = H \text{ for } i = j, j+1, ..., n.$$

The Shishkin mesh makes Backward Euler errors uniform in $\varepsilon$. The following can be proved.

**Theorem 4.1** *The Backward Euler method (14) on a Shishkin mesh gives errors*

$$|u(x_i) - y_i| \leq C\ \frac{\ln(n)}{n}, \qquad i = 1, 2, ..., n$$

*Proof: Refer to [4], [5], and [7].*

## 5 Final Numerical Experiments

We first examine the method with the previous test problem (15) with parameters $w = a_* = \omega = 1$, $j = \frac{n}{2}$.

Table 3: Backward Euler Results for (15) on Shishkin Mesh with parameters $w = a_* = \omega = 1$, $j = \frac{n}{2}$

| $\varepsilon$ | n = 10 | 20 | 40 | 80 | 160 |
|---|---|---|---|---|---|
| 1 | .02725 | .01399 | .00709 | .00357 | .00179 |
| .1 | .69761 | .48149 | .31066 | .19016 | .11058 |
| .01 | .70682 | .48871 | .31505 | .19266 | .11369 |
| $10^{-3}$ | .70691 | .48879 | .31510 | .19269 | .11371 |
| $10^{-4}$ | .70691 | .48879 | .31510 | .19269 | .11371 |
| $10^{-5}$ | .70691 | .48879 | .31510 | .19269 | .11371 |
| $10^{-6}$ | .70691 | .48879 | .31510 | .19269 | .11371 |

We now examine the method with a new problem (17) in which $\omega > a_*$.

$$\varepsilon u' + e^{-x}u = 0, \qquad x \in (0,1), \qquad u(0) = 1, \tag{17}$$

with solution

$$u = e^{\frac{1}{\varepsilon}(e^{-x}-1)}.$$

Now $\tilde{a}(x) = e^{\frac{1}{x}}$ since (17) is linear, and we have $a(x,u) = \tilde{a}(x)u + \tilde{b}(x) = 0$. Therefore, $\omega = 1$ by (12), and $a_* = \frac{1}{e}$ by (2). Table 2 shows the results with parameters $w = \omega = 1$, $j = \frac{n}{2}$.

Table 4: Backward Euler Results for (17) on Shishkin Mesh with parameters $w = \omega = 1$, $j = \frac{n}{2}$

| $\varepsilon$ | n = 10 | 20 | 40 | 80 | 160 |
|---|---|---|---|---|---|
| 1 | .02669 | .01375 | .00697 | .00351 | .00176 |
| .1 | .07428 | .05150 | .03315 | .02025 | .01177 |
| .01 | .07104 | .04913 | .03167 | .01936 | .01142 |
| $10^{-3}$ | .07072 | .04890 | .03152 | .01927 | .01137 |
| $10^{-4}$ | .07069 | .04888 | .03151 | .01927 | .01137 |
| $10^{-5}$ | .07069 | .04887 | .03151 | .01926 | .01137 |
| $10^{-6}$ | .07069 | .04887 | .03151 | .01926 | .01137 |

Juxtapose these results with the errors given by the same method on (17) with parameters $w = a_* = \frac{1}{e}$, $j = \frac{n}{2}$. See Table 5.

Table 5: Backward Euler Results for (17) on Shishkin Mesh with parameters $w = a_* = \frac{1}{e}$, $j = \frac{n}{2}$

| $\varepsilon$ | n = 10 | 20 | 40 | 80 | 160 |
|---|---|---|---|---|---|
| 1 | .02669 | .01375 | .00697 | .00351 | .00176 |
| .1 | .13886 | .08047 | .04385 | .02298 | .01177 |
| .01 | .15895 | .10877 | .07714 | .04890 | .02979 |
| $10^{-3}$ | .15818 | .10828 | .07679 | .04867 | .02965 |
| $10^{-4}$ | .15810 | .10823 | .07676 | .04865 | .02964 |
| $10^{-5}$ | .15810 | .10823 | .07676 | .04865 | .02964 |
| $10^{-6}$ | .15810 | .10823 | .07676 | .04865 | .02964 |

Also, compare the results of Table 4 to those in Table 6 which were obtained by choosing $j = \lceil \frac{3n}{4} \rceil$. This makes the mesh denser in the layer.

Table 6: Backward Euler Results for (17) on Shishkin Mesh with parameters $w = \omega = 1$, $j = \lceil \frac{3n}{4} \rceil$

| $\varepsilon$ | n = 10 | 20 | 40 | 80 | 160 |
|---|---|---|---|---|---|
| 1 | .02634 | .01443 | .00754 | .00351 | .00194 |
| .1 | .04983 | .03568 | .02261 | .00385 | .00791 |
| .01 | .04757 | .03412 | .02162 | .01370 | .00768 |
| $10^{-3}$ | .04735 | .03396 | .02152 | .01310 | .00764 |
| $10^{-4}$ | .04733 | .03395 | .02151 | .01304 | .00764 |
| $10^{-5}$ | .04733 | .03395 | .02151 | .01304 | .00764 |
| $10^{-6}$ | .04733 | .03395 | .02151 | .01304 | .00764 |

Finally, the results with parameters $w = a_* = \frac{1}{e}, j = \lceil \frac{3n}{4} \rceil$. See Table 7.

Table 7: Backward Euler Results for (17) on Shishkin Mesh with parameters $w = a_* = \frac{1}{e}$, $j = \lceil \frac{3n}{4} \rceil$

| $\varepsilon$ | n = 10 | 20 | 40 | 80 | 160 |
|---|---|---|---|---|---|
| 1 | .02634 | .01443 | .00754 | .00385 | .00176 |
| .1 | .09727 | .05675 | .03015 | .01557 | .00194 |
| .01 | .10632 | .08285 | .05440 | .03392 | .00791 |
| $10^{-3}$ | .10570 | .08248 | .05416 | .03377 | .02020 |
| $10^{-4}$ | .10564 | .08244 | .05413 | .03375 | .02019 |
| $10^{-5}$ | .10563 | .08243 | .05413 | .03375 | .02019 |
| $10^{-6}$ | .10563 | .08243 | .05413 | .03375 | .02019 |

These results show that the error is uniform in $\varepsilon$ for the Backward Euler method on a Shishkin mesh. This can be observed in the fact that as epsilon decreases, the errors remain the same. Remark this is not the case on the uniform mesh; see Table 2. At the same time, we can see that the errors decrease when $n$ increases which agrees with Theorem 4.1.

To improve the results on the Shishkin mesh I changed the density of the mesh in the layer. I did this by increasing the number of mesh steps $j$ in the layer, and by choosing the greater value for the parameter $w$. The latter was possible to do in the second test problem (17). In particular, Table 6 contains the smallest errors, and this is where the mesh is the most dense in the layer. Thus we have improvement of the result in [4] along the lines of [5].

The improvements of the Shishkin mesh were enabled by the results from Lemma 2.9 and Theorem 2.10. These results are the new contribution of the thesis.

## References

[1] K.W. Chang & F.A. Howes (1984). *Nonlinear Singular Perturbation Phenomena: Theory and Application, Applied Mathematical Sciences, vol. 56*, Springer: New York.

[2] Hermann, M., & Saravi, M. (2014). *A First Course in Ordinary Differential Equations: Analytical and Numerical Methods, Springer: New York*.

[3] Burden, R. L. & Faires, J. D. (2011). *Numerical Analysis.* Boston: Richard Stratton.

[4] R. Vulanović (1991). *A Second Order Numerical Method for Nonlinear Singular Perturbation Problems Without Turning Points*, *Zh. Vychisl. Mat. i Mat. Fiz.* 31, 522–532.

[5] R. Vulanović & Lj. Teofanov (2014). *On the Quasilinear Boundary-Layer Problem and its Numerical Solution*, *J. Comput. Appl. Math.* 268, 56–67.

[6] Protter, M. H., & Weinberger, H. F. (1967). *Maximum Principles in Differential Equations.* Englewood Cliffs, NJ: Prentice-Hall.

[7] R. Vulanović (1989). *A Uniform Numerical Method for Quasilinear Singular Perturbation Problems Without Turning Points*, *Computing* 41, 97–106.

## A Python - Backward Euler on Shishkin Mesh

```
"""
|==========================================================================|
| Author: Jake Adkins                                                            |
| Advisor: Dr. Relja Vulanovic                                                   |
| ---Kent State University at Stark, Canton - OH                                 |
| Project: A Robust Numerical Method for a Singularly Perturbed Nonlinear       |
| ---Initial Value Problem                                                       |
| Date: 9/19/2017                                                                |
|================================================================================|
"""

import sys
from math import log, exp, ceil

#Initialize Global Variables
eps = 10.0;
EPS_ORDER = 7;

#Lexical Function Declarations
def u(current_x): #Continuous solution to the Problem
return exp((1/eps)*(exp(-current_x) - 1));

def til_a(x):
return exp(-x);

def til_b(x):
return 0;

def main():

    #Declare Global Variables
    global eps;
    global EPS_ORDER;

    #Declare / Define local variables
    #N - total number of steps in the Shishkin Mesh
    x0 = 0.0; # initial x value
    y0 = 1.0; # initial y value
    x = 1.0; # width (x)
    err = 0.0; # measures Maximum Absolute Error for each iteration
    w = 1.0/exp(1);
    #J - Compare to j;
    #y - numerical solution
    #h - step size in the fine mesh
    #tau - Transition point
    #H - step size in the crude mesh

    N = int(input('N: '));
    J = ceil(3*N/4.0);

    for i in range(0, EPS_ORDER): # loop for modifying algorithm params
        eps =  eps / 10.0;
        print("Epsillon: " + str(eps));

        tau = min(0.5, (1.0 * eps / w)*log(N));

        H = (1 - tau) / (N-J);

        h = tau/J;

        for j in range(1, N):

            if j <= J: #if x0 <= tau: # determine which mesh we are in
                active_mesh_step = h;
            else:
                active_mesh_step = H;

            y = (eps * y0 - (active_mesh_step * til_b(x0 + active_mesh_step)))/(eps + (til_a
            err = max(err, abs(y - u(x0 + active_mesh_step)));

            y0 = y;
            x0 = x0 + active_mesh_step;

            #Report Progress
            if j == N - 1:
                print("Maximum Error: %.20f" % err);
                print('');

        #Reset Parameters
        err = 0.0;
        x0 = 0.0;
        y0 = 1.0;

    return 0;

if __name__ == "__main__":
    main()
```

## B C++ - Backward Euler

```
#include <iostream>
#include <iomanip>
```

```
#include <algorithm>
#include <fstream>
using namespace std;

double u(double x);

double eps = .1;
const int EPS_ORDER = 0;

int main()
{

    int n;
    double x0, y0, x, y, h;
    double error = 0;
    ofstream out;
    out.open("out.txt");

    out.setf(ios::fixed);
    out << "Enter the initial values of x and y respectively: 0 10\n"; //Initial values
    x0 = 0; y0 = 10;
    out << "Width of interval: 1" << endl;
    x = 1;
    cout << "Enter the width of the sub-interval: ";
    cin >> h;

    out << "x" << setw(19) << "y" << setw(19) << "Error" << endl;
    out << "-------------------------------------------------------\n";
    for (int i = 0; i <= EPS_ORDER; i++) { //Ininitialize epsilon
        eps /= 10;
        out << "Epsilon: " << eps << endl;

        //while(x0<x)
        while (fabs(x - x0) > 0.0000001)
        {
            y = (eps / (eps + h))*y0 - (h / (eps + h))*(-10 - exp(-(x0 + h))*(eps*(9 + (x0 -
            error = max(error, abs(y - u(x0+h)));
            y0 = y;                    //pass this new y as y0 in the next iteration.
            x0 = x0 + h;                //calculate new x.

out << u(x0 + h) << endl;
        }

        out.precision(20);
        out << "Error: " << error << endl;
        out << "Step: " << h << endl << endl;
        //reset
        error = 0;
        x0 = 0;
        y0 = 10;
    }

    system("PAUSE");
    return 0;
}

double u(double x)
{
    return 10 * (1 + exp(-x / eps)) - (10 + x)*exp(-x);
}
```