# DEVELOPING AND IMPLEMENTING TWO-STEP ADAMS-BASHFORTH-MOULTON METHOD WITH VARIABLE STEPSIZE FOR THE SIMULATION TOOL DYNSTAR

**An Pletinckx**[1]; **Daniel Fiß**[2]; **Alexander Kratzsch**[3]
Hochschule Zittau/Görlitz, IPM Department
Theodor-Körner-Allee 16, 02763 Zittau, Germany
e-mail: [1]an.pletinckx@vub.ac.be; [2]d.fiss@hszg.de; [3]a.kratzsch@hszg.de

## Abstract

The simulation tool DynStar, created by Hochschule Zittau/Görlitz IPM department, was previously using only single-step methods to solve differential equations. This paper describes the development of a multiple step method to complement the others. The Introduction gives the reader a better idea why a multiple step method can be useful. The theory part is focused on the two-step Adams-Bashforth-Moulton method and how it is possible to make the formulas suitable for variable stepsize. Further, an algorithm is developed to solve the differential equations, using the ABM formulas and adjusting the stepsize according to the error between the prediction and the correction. This is described in the Implementation section. Finally, the performance of the ABM method is compared with RK4 and the Hanna method in the Results section.

## Keywords

Adams-Bashforth-Moulton; Multistep method; Variable stepsize; Numerical solution; Ordinary differential equation; Initial-value problem; Hanna method.

## Introduction

In engineering, it is often necessary to simulate the behaviour of real-life systems. For this, a mathematical model that describes the system is needed. When taking into account the speed of some variations, the description may contain derivatives. This means that you will end up with one or more differential equations in the model. Then you are left with the problem of solving these equations, which can be done analytically or numerically. In the era of computers, numerical solving techniques have become increasingly important.

Considering an initial-value problem $y' = f(x, y)$ and $y(x_0) = y_0$, many methods have been studied to solve this differential equation numerically [1]. The simplest one is undoubtedly the (forward) Euler method, where the following value $y_{n+1}$ is calculated as:

$$y_{n+1} = y_n + h \cdot f(x_n, y_n) \tag{1}$$

Of course, such a formula cannot be used in practice because it generally will not give an accurate result. An attempt to construct a more reliable method has led to the improved Euler method, which uses the average of the slopes in $(x_n, y_n)$ and $(x_{n+1}, y_{n+1})$. Hence $y_{n+1}$ is calculated implicitly:

$$y_{n+1} = y_n + \frac{h}{2}(f(x_n, y_n) + f(x_{n+1}, y_{n+1})) \tag{2}$$

Solving an implicit equation on its own can be difficult because it leads to a non-linear problem. However, this difficulty can be avoided by combining the implicit formula with an

explicit one in a predictor-corrector scheme; in this case the forward Euler method would be the ideal choice. Unfortunately, the accuracy of the improved Euler method is still disappointing.

Finally, generalization of the Euler formulas was created, which is now known as the Runge-Kutta method. It uses a weighted average of slopes in the interval $x_n \leq x \leq x_{n+1}$.

$$y_{n+1} = y_n + h(w_1 k_1 + w_2 k_2 + \cdots + w_m k_m) \tag{3}$$

It can be concluded that the Euler method and the improved Euler method are in fact Runge-Kutta methods of the first and second order respectively. The formulas that are most commonly used, because they are very accurate and still easy to implement, are of the fourth order.

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4}$$

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot k_1\right) \\ k_3 &= f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot k_2\right) \\ k_4 &= f(x_n + h, y_n + h \cdot k_3) \end{aligned}$$

Even though RK4 is popular for its accuracy, it has the major disadvantage that in each step the function must be calculated four times. When function evaluations are expensive, this can seriously prolong the runtime. A solution is found in the category of multistep methods. Notice that the methods mentioned above are all single-step methods because in order to compute the successive value $y_{n+1}$, information about only one prior value $y_n$ is needed. Multistep methods use several points in the calculation, but they gain efficiency by storing the previous information such that only one new function evaluation has to be computed at each step. To improve the accuracy and stability of the algorithm, one can choose to add a corrector step which also requires an additional function evaluation. This doubles the cost, but it is still an improvement over the Runge-Kutta method. The best-known multistep method is the Adams-Bashforth-Moulton one, which will be discussed in the next section.

## 1 Theory

Adams-Bashforth and Adams-Moulton are linear multistep methods. This means that the next value $y_{n+1}$ is calculated as a linear combination of various $y_i$ and $f(x_i, y_i)$ from the previous $s$ steps:

$$\begin{aligned} y_{n+1} + a_1 y_n + a_2 y_{n-1} + \cdots + a_s y_{n-s+1} \\ = h\big(b_0 f(x_{n+1}, y_{n+1}) + b_1 f(x_n, y_n) + \cdots + b_s f(x_{n-s+1}, y_{n-s+1})\big) \end{aligned}$$

For Adams-Bashforth, coefficient $a_1 = -1$ and all others $a_2 = \cdots = a_s = 0$. The coefficients $b_i$ are derived by considering the following form:

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f\big(t, y(t)\big) \cdot dt \approx y(x_n) + \int_{x_n}^{x_{n+1}} p(t) \cdot dt \tag{5}$$

It's possible to replace the function $f(t, y(t))$ by the interpolation polynomial $p(t)$ through the s previous points [2]. After the polynomial is calculated, an expression for the coefficients $b_i$ can be found. For example, the two step Adams-Bashforth formula becomes:

$$y_{n+1} = y_n + h \cdot (\frac{3}{2} f_n - \frac{1}{2} f_{n-1}) \quad (6)$$

However, the integration polynomial was integrated from $x_n$ to $x_{n+1}$, while its interpolation interval is limited to $[x_{n-s+1}, x_n]$. In general, integration polynomials cannot be used as a reliable approximation outside their interpolation interval. So it is logical to include the point $(x_{n+1}, y_{n+1})$ in the polynomial and do the calculations again. This results in the implicit Adams-Moulton formulas. Again the two-step formula is shown:

$$y_{n+1} = y_n + h \cdot (\frac{5}{12} f_{n+1} + \frac{8}{12} f_n - \frac{1}{12} f_{n-1}) \quad (7)$$

Adams-Moulton has some great advantages over the explicit Adams-Bashforth [3]. As already mentioned, Adams-Moulton methods give more accurate approximations due to the wider interpolation interval. They also obtain a higher order with the same amount of previous steps and generally, implicit methods are more stable than their explicit counterparts. Of course, the drawback lies in its implicit nature, which makes it difficult to solve as it translates to a non-linear equation.

The way to profit from both methods is to combine them in a predictor-corrector scheme. The explicit Adams-Bashforth method computes a prediction $y^*_{n+1}$. At this point, the function evaluation $f^*_{n+1}$ is calculated. Then the corrected value $y_{n+1}$ is obtained with the Adams-Moulton method and again, the function is evaluated to use in later steps. This is known as the PECE procedure, but it is also possible to repeat the correction step, for example PECECE or even more. Each additional correction step C makes the result more accurate, but also introduces a new function evaluation E, which partly or completely reverses the advantage that ABM has over RK4. A compromise is made by performing the correction step only once while still ensuring a sufficient accuracy. This is established by starting the calculation over with a smaller stepsize when the correction differs too much from the prediction.

This includes that we are now facing a variable stepsize problem and that formulas (6) and (7) must be adapted, because they assume a constant stepsize. The coefficients $b_i$ depend on the ratio of the stepsizes. That's why they are constant for a fixed stepsize, but at each step they need to be calculated in a variable stepsize problem. There are different ways to make the Adams methods suitable for variable stepsizes, but we will follow a procedure developed by Krogh and described by Lopez and Romay in their paper [4].

The expression of the variable stepsize *k*-step Adams-Bashfort predictor and Adams-Moulton corrector is as follows:

$$p_{n+1} = y_n + h_n \sum_{j=0}^{k-1} g_j(n)\, \beta_j(n)\, \phi_j(n) \quad (8)$$

$$y_{n+1} = p_{n+1} + h_n\, g_k(n)\, \phi_k(n+1) \quad (9)$$

$$\text{where } h_n = x_{n+1} - x_n$$

The coefficients are defined recursively:

$$\beta_j(n) = \beta_{j-1}(n)\frac{x_{n+1} - x_{n-j+1}}{x_n - x_{n-j}} \quad \text{with} \quad \beta_0(n) = 1 \tag{10}$$

$$\phi_j(n) = \phi_{j-1}(n) - \beta_{j-1}(n-1)\phi_{j-1}(n-1) \quad \text{with} \quad \phi_0(n) = f_n \tag{11}$$

$$g_j(n) = c_{j,1}(x_{n+1}) \tag{12}$$

$$c_{j,q}(x_{n+1}) = c_{j-1,q}(x_{n+1}) - c_{j-1,q+1}(x_{n+1})\frac{h_n}{x_{n+1} - x_{n-j+1}} \quad \text{with} \quad c_{0,q}(x_{n+1}) = \frac{1}{q} \tag{13}$$

## 2 Implementation

Here, all this theory should be put into practice and implemented into the DynStar source code. The most important step is to derive the correct formulas. The two-step ABM method has been chosen because its formulas are the easiest to derive and implement, but the code can be extended to allow other ABM methods with more complicated formulas. For $k = 2$, the formulas are derived as follows:

$$p_{n+1} = y_n + h^+(g_0(n)\,\beta_0(n)\,\phi_0(n) + g_1(n)\,\beta_1(n)\,\phi_1(n))$$

$$\Longrightarrow p_{n+1} = y_n + h^+\left(f_n + \frac{1}{2}\frac{h^+}{h^-}(f_n - f_{n-1})\right) \tag{14}$$

$$\text{and} \quad y_{n+1} = p_{n+1} + h^+\, g_2(n)\,\phi_2(n+1)$$

$$\Longrightarrow y_{n+1} = p_{n+1} + h^+\left(\frac{1}{2} - \frac{1}{6}\frac{h^+}{h^+ + h^-}\right)\left(f_{n+1} - f_n - \frac{h^+}{h^-}(f_n - f_{n-1})\right) \tag{15}$$

where $h^- = x_n - x_{n-1}$ is the previous stepsize and $h^+ = x_{n+1} - x_n$is the following stepsize. Notice that if the stepsizes are equal, the formulas simplify to (6) and (7).

It's clear that the k-step ABM method needs information about k previous points in order to calculate the following value. However, in the beginning of the algorithm only one value is available: the initial condition $y(x_0) = y_0$. Hence the ABM method is not self-starting and another method is needed to calculate the k-1 first values. In our case, only one extra value is required and it is computed using one step of Runge-Kutta fourth order.

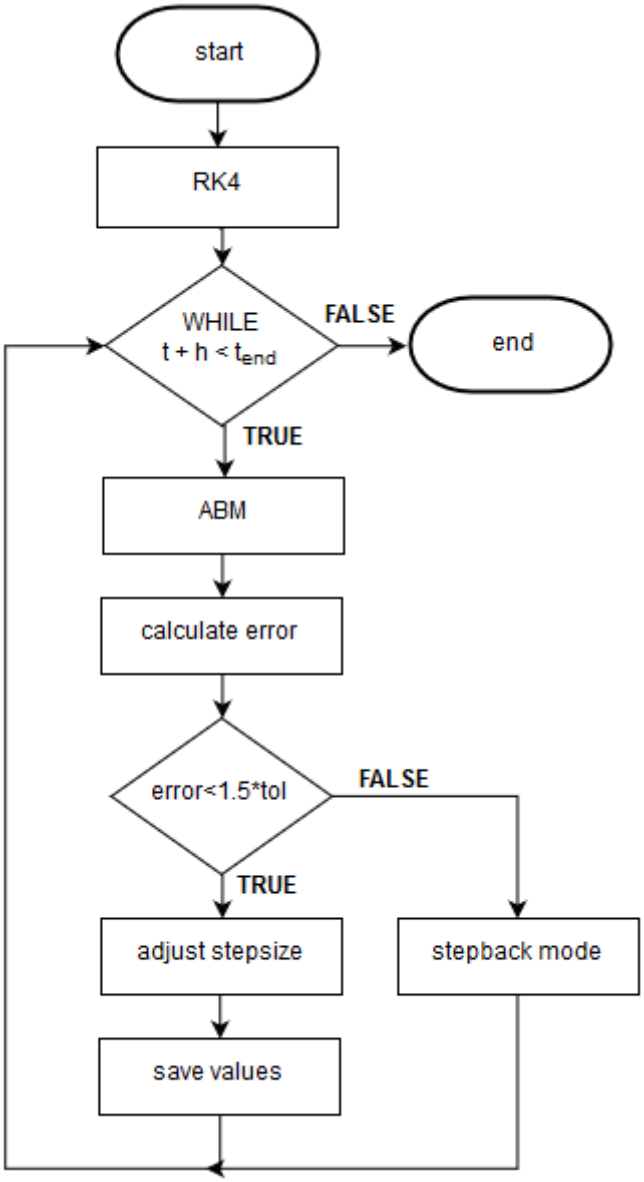
The next feature that needs to be implemented is the varying stepsize. The reasoning behind it is that the stepsize should always be small enough to ensure accurate results, yet large enough to avoid unnecessary calculations. The local truncation error can be used as a measure for accuracy. This error is estimated as the relative difference between the predicted and the corrected value.

$$\text{relative error} = \text{abs}\left(\frac{p_{n+1} - y_{n+1}}{y_{n+1}}\right) \tag{16}$$

When this error gets too large, i.e. when it exceeds a certain tolerance, the stepsize should be decreased immediately. On the other hand, when the error is below the tolerance, it is safe to increase the stepsize. With these properties in mind, Hanna [5] proposed a formula to adjust the stepsize:

$$h^+ = h^-\sqrt{\frac{tol}{rel}} \tag{17}$$

However, sometimes the error is unacceptably high, quantified by exceeding 1.5 times the tolerance. In this case, choosing a small stepsize for the following step is not enough; it is the stepsize for the current step that needs to be decreased. Then the program goes into a ‘stepback mode’, where none of the new values are saved and the current step is completely restarted with a smaller stepsize. The flowchart in Figure 1 shows how the algorithm is organized.




*Source: Own*

***Fig. 1:*** *Flowchart of the algorithm*

## 3 Results

### 3.1 ABM Compared to RK4

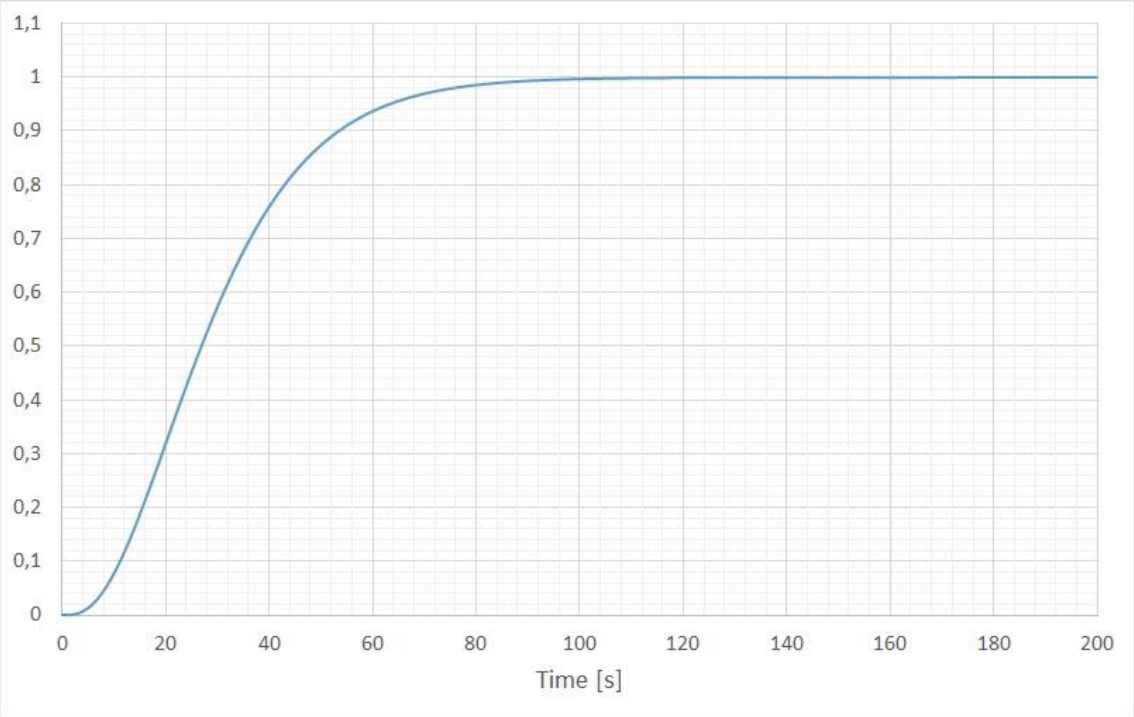
First, a comparison between ABM and RK4 is made. The methods are tested on a very simple third-order ordinary differential equation given by (18). The solution is shown in Figure 2.

$$y'_1 = \frac{1}{T_1}(u - y_1) \quad \text{with} \quad y_1(0) = 0$$

$$y'_2 = \frac{1}{T_2}(y_1 - y_2) \quad \text{with} \quad y_2(0) = 0 \tag{18}$$

$$y'_3 = \frac{1}{T_3}(y_2 - y_3) \quad \text{with} \quad y_3(0) = 0$$

$$t_0 = 0 \quad \text{and} \quad t_{end} = 200$$




*Source: Own*

***Fig. 2:*** *Graph of PT3*

As ABM calculates only two function evaluations per integration step and RK4 calculates four, one would expect ABM to be twice as fast. However, in table 1 it can be seen that the results do not support this hypothesis. In fact, the amount of time needed by the two methods is approximately the same. The explanation is that the function evaluation is very simple or cheap and does not account for a large portion of the runtime. Therefore, it can almost be neglected whether the evaluation is done two or four times. But when the differential equation is computationally expensive, the amount of function evaluations will certainly be reflected in the runtime. To demonstrate this, we still use the same problem given by (18) but we make it expensive by adding a delay of 1ms. Now it is clear that Runge-Kutta takes twice as much time as ABM. In all tables, the time is expressed in seconds.

***Tab. 1:*** *ABM compared to RK4*

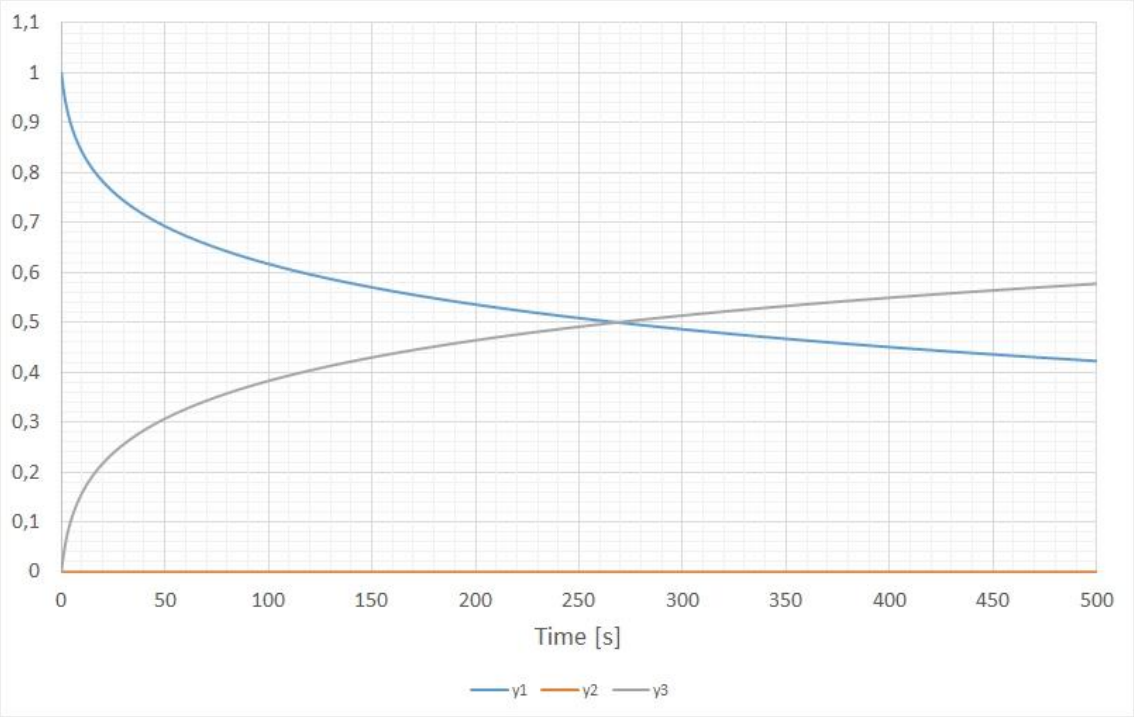
| TOL | Steps | | Time (cheap evaluation) | | Time (expensive evaluation) | |
|---|---|---|---|---|---|---|
| | ABM | RK4 | ABM | RK4 | ABM | RK4 |
| $10^{-4}$ | 116 | 121 | 0.02 | 0.02 | 0.46 | 0.95 |
| $10^{-7}$ | 1014 | 1037 | 0.14 | 0.15 | 3.99 | 8.12 |

*Source: Own*

### 3.2 ABM Compared to Hanna, Stiff Problem

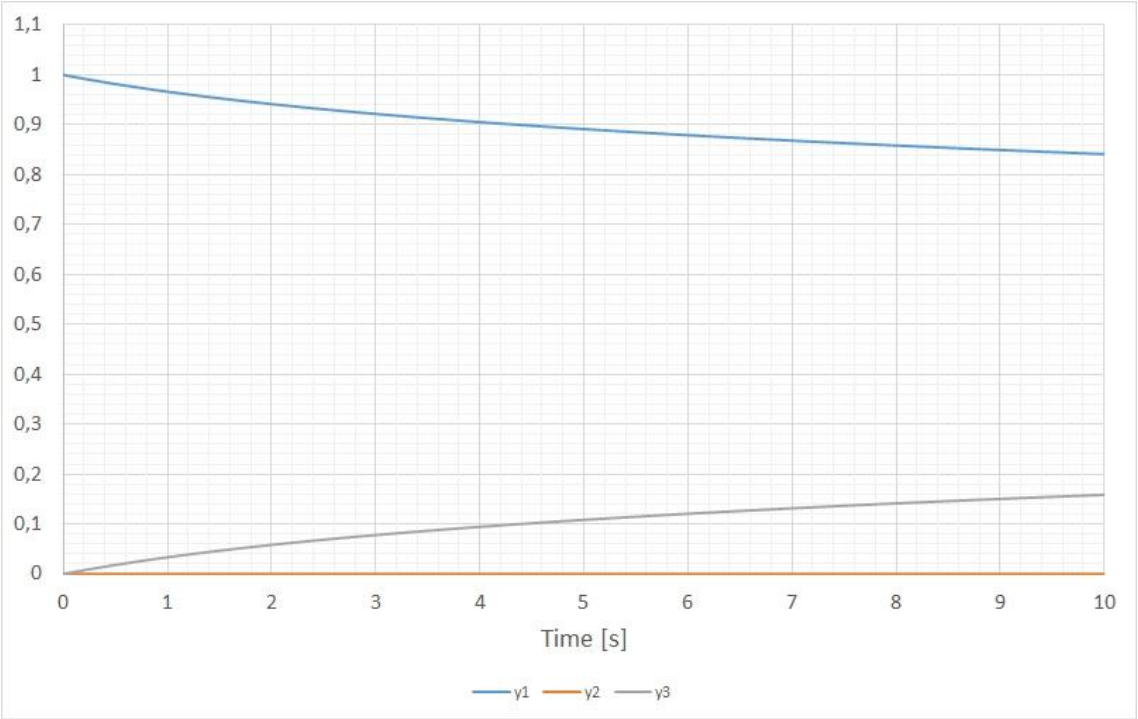
The method that has already been implemented in DynStar is the one Hanna proposed in his paper [5]. It is important to understand which method is more effective for different cases, so that a DynStar user can choose the right option. Hanna method takes a weighted average of the values obtained by the Euler method and the RK2 (or improved Euler) method. The great advantage of the Hanna method is that it has a large stability domain. For stiff problems the stepsize is limited rather for stability reasons than accuracy reasons. This means that a stable method such as Hanna will be able to take bigger steps, hence saving time. Unfortunately, ABM doesn't handle stiff problems very well and is more suitable for non-stiff problems. We will test this hypothesis on a mildly stiff problem, also discussed by Hanna in his paper, called Robertson chemical reaction system, given by (19). In Figures 3 and 4, the solution is shown for different end times. Both graphs are in linear scale.

$$
\begin{aligned}
y'_1 &= -0.04 \cdot y_1 + 10^4 \cdot y_2 y_3 \quad \text{with} \quad y_1(0) = 1 \\
y'_2 &= 0.04 \cdot y_1 - 10^4 \cdot y_2 y_3 - 3 \cdot 10^7 {y_2}^2 \quad \text{with} \quad y_2(0) = 0 \\
y'_3 &= 3 \cdot 10^7 {y_2}^2 \quad \text{with} \quad y_3(0) = 0 \\
t_0 &= 0 \quad \text{and} \quad t_{end} = 10
\end{aligned}
\tag{19}
$$




*Source: Own*

***Fig. 3:*** *Graph of Robertson chemical reaction system, t: 0 ⟶ 500 seconds*




*Source: Own*

***Fig. 4:*** *Graph of Robertson chemical reaction system, t: 0 ⟶ 10 seconds*

The results in Table 2 show that ABM is forced to take small steps in order to maintain stability, while achieving high accuracy in the process. On the other hand, Hanna is much faster and in many cases this is preferred. Only when tolerances are very strict, Hanna gets in trouble because it cannot be used to obtain highly accurate results. In the following table, RE is the relative error between the computed solution for $y_3$ at $t = 10$ and the exact solution $y_3(10) = 0.1586138397$.

***Tab. 2:*** *ABM compared to Hanna in case of a moderately stiff problem*

| | RE | | Steps | | Time | |
|---|---|---|---|---|---|---|
| TOL | ABM | HAN | ABM | HAN | ABM | HAN |
| $10^{-2}$ | $2.8 \cdot 10^{-6}$ | $1.2 \cdot 10^{-4}$ | 13696 | 3931 | 1.34 | 0.28 |
| $10^{-3}$ | $3.4 \cdot 10^{-8}$ | $1.8 \cdot 10^{-4}$ | 14022 | 4083 | 1.42 | 0.35 |
| $10^{-4}$ | $1.5 \cdot 10^{-8}$ | $1.3 \cdot 10^{-4}$ | 14060 | 4550 | 1.37 | 0.37 |
| $10^{-5}$ | $1.6 \cdot 10^{-8}$ | $1.2 \cdot 10^{-4}$ | 14159 | 5284 | 1.31 | 0.39 |
| $10^{-6}$ | $1.6 \cdot 10^{-8}$ | $8.9 \cdot 10^{-5}$ | 14475 | 8597 | 1.43 | 0.81 |
| $10^{-7}$ | $1.6 \cdot 10^{-8}$ | $5.8 \cdot 10^{-5}$ | 14753 | 19397 | 1.54 | 1.90 |
| $10^{-8}$ | $1.6 \cdot 10^{-8}$ | $2.6 \cdot 10^{-5}$ | 15733 | 54924 | 1.71 | 4.92 |

*Source: Own*

### 3.3 ABM Compared to Hanna, Non-stiff Problem

Further, the performances of these two methods are compared for a non-stiff problem. We use a differential equation system similar to (18). The difference is that now we use a tenth order ODE given by (20) and all time constants are chosen to be equal, ensuring a non-stiff problem. The solution is shown in Figure 5.

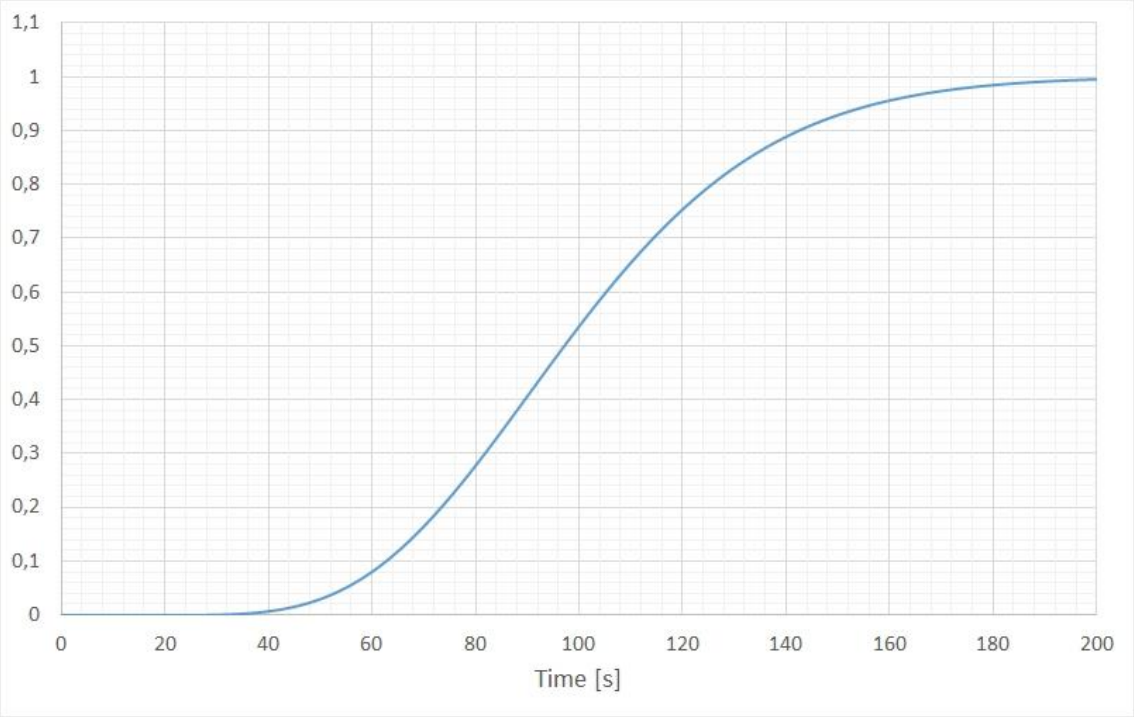
$$y'_1 = \frac{1}{T}(u - y_1) \quad \text{with} \quad y_1(0) = 0$$

$$y'_2 = \frac{1}{T}(y_1 - y_2) \quad \text{with} \quad y_2(0) = 0$$

$$\vdots \tag{20}$$

$$y'_{10} = \frac{1}{T}(y_9 - y_{10}) \quad \text{with} \quad y_{10}(0) = 0$$

$$t_0 = 0 \quad \text{and} \quad t_{end} = 200$$




*Source: Own*

***Fig. 5:*** *Graph of PT10*

Now the results in table 3 are very positive for ABM because this method is both faster and more accurate than the Hanna method. This can be explained by remembering that for a non-stiff problem, stability is not much of an issue. This means that an accurate method such as ABM can allow for taking bigger steps than a stable method such as Hanna, hence needing fewer steps and less time. In table 3, RE is defined as the relative error between the computed solution for $y_{10}$ at $t = 200$ and the exact solution $y_{10}(200) = 0.9947057955$.

***Tab. 3:*** *ABM compared to Hanna in case of a non-stiff problem*

| TOL | RE | | Steps | | Time | |
|---|---|---|---|---|---|---|
| | ABM | HAN | ABM | HAN | ABM | HAN |
| $10^{-2}$ | $3.5 \cdot 10^{-3}$ | $5.8 \cdot 10^{-3}$ | 1279 | 1621 | 0.12 | 0.17 |
| $10^{-3}$ | $3.8 \cdot 10^{-4}$ | $3.0 \cdot 10^{-3}$ | 1623 | 2942 | 0.21 | 0.52 |
| $10^{-4}$ | $4.0 \cdot 10^{-5}$ | $1.1 \cdot 10^{-3}$ | 2362 | 7120 | 0.39 | 1.46 |
| $10^{-5}$ | $4.0 \cdot 10^{-6}$ | $3.8 \cdot 10^{-4}$ | 3954 | 20326 | 0.82 | 4.41 |
| $10^{-6}$ | $4.2 \cdot 10^{-7}$ | $1.2 \cdot 10^{-4}$ | 7383 | 62087 | 1.68 | 14.39 |

*Source: Own*

## Conclusion

The goal of the assignment was to improve the simulation tool DynStar by adding another solver algorithm for differential equations. After literature research, we succeeded to develop formulas for a multistep method with a variable stepsize, namely the two-step Adams-Bashforth-Moulton predictor-corrector method as described by Krogh. The method has been tested and the following conclusions can be drawn. ABM is most effective when function evaluations are expensive and/or when the problem is non-stiff. On the other hand, when function evaluations are cheap, ABM still presents good, but not significantly better results than RK4. In the case of a stiff problem, the Hanna method is more efficient in terms of runtime; however, the ABM method gets more accurate results. Altogether, we can conclude that the ABM algorithm is a valuable addition to the DynStar software and that it will prove itself useful by solving many differential equations more accurately and/or more efficiently than the methods that have already been implemented.

## Literature


[1] ZILL, D. G.; WRIGHT, W. S.: *Advanced Engineering Mathematics, Fourth edition*, Jones & Bartlett Learning, Massachusetts, 2011, pp. 275–286.

[2] HAIRER, E.; NORSETT, S. P.; WANNER, G.: *Solving Ordinary Differential Equations I: Nonstiff Problems, Second revised edition*, Springer, Berlin, 2008, pp. 357–360.

[3] BUTCHER, J. C.: Numerical Methods for Ordinary Differential Equations in the 20th Century, *Journal of Computational and Applied Mathematics*, 2000, Vol. 125, Issues 1–2, pp. 1–29.

[4] LÓPEZ, D. J.; ROMAY, J. G.: Implementing Adams Methods with Preassigned Stepsize Ratios, *Mathematical Problems in Engineering*, 2010, DOI: 10.1155/2010/765620

[5] ASHOUR, S. S.; HANNA, O. T.: A New Very Simple Explicit Method for the Integration of Mildly Stiff Ordinary Differential Equations, *Computers and Chemical Engineering*, 1990, Vol. 14, Issue 3, pp. 267–272.

## ROZVOJ A REALIZACE DVOUKROKOVÉ ADAMS-BASHFORTH-MOULTONOVY METODY S VARIABILNÍ VELIKOSTÍ KROKŮ U SIMULAČNÍHO NÁSTROJE DYNSTAR

Simulační nástroj DynStar, vytvořený na katedře IPM na Hochschule Zittau / Görlitz, dříve používal pro řešení diferenciálních rovnic pouze jednokrokové metody. Tento článek popisuje vývoj vícekrokové metody doplňující ostatní metody. Úvodní část dává čtenáři lepší představu o užitečnosti vícekrokové metody. Teoretická část je zaměřena na dvoukrokovou Adams-Bashforth-Moultonovu metodu (ABM), a možnost vhodného využití vzorců pro variabilní velikost kroků (stepsize). Poté je vytvořen algoritmus řešení diferenciálních rovnic za použití vzorců ABM a úpravou stepsize podle odchylky mezi predikcí a korekcí. To je popsáno v sekci Realizace. V závěrečné sekci článku (Výsledky) je metoda ABM porovnána s metodami RK4 a Hanna.

## ENTWICKLUNG UND IMPLEMENTIERUNG DES ZWEI-SCHRITT-ADAMS-BASHFORTH-MOULTON-VERFAHREN MIT VARIABLER SCHRITTWEITE FÜR DAS SIMULATIONSWERKZEUG DYNSTAR

Das Simulationswerkzeug DynStar, das von der Hochschule Zittau / Görlitz IPM entwickelt wird, verwendet bisher nur einstufige Methoden zur Lösung von Differentialgleichungen. Dieser Beitrag beschreibt die Entwicklung eines mehrstufigen Verfahrens. Die Einleitung gibt legt dar, warum eine mehrstufige Methode nützlich sein kann. Der Theorieteil konzentriert sich auf das zweistufige Adams-Bashforth-Moulton (ABM)-Verfahren und wie es für variable Schrittweiten angepasst wurde. Es wurde ein Algorithmus entwickelt, um die Differentialgleichungen unter Verwendung des ABM-Methode zu lösen und die Schrittweite entsprechend dem Fehler zwischen der Vorhersage und der Korrektur zu bestimmen. Dies wird im Implementierungsabschnitt beschrieben. Abschließend wird die ABM-Methode mit dem Runge-Kutta-Verfahren 4. Ordnung und der Hanna-Methode im Ergebnisabschnitt verglichen.

## OPRACOWANIE I REALIZACJA DWUKROKOWEJ METODY ADAMS-BASHFORTH-MOULTONA ZE ZMIENNĄ WIELKOŚCIĄ KROKÓW W NARZĘDZIU SYMULACYJNYM DYNSTAR

Narzędzie symulacyjne DynStar, opracowane w katedrze IPM (procesów automatyzacyjnych i technik pomiarowych) w Hochschule Zittau/Görlitz, do rozwiązywania równań różniczkowych wykorzystywało dawniej wyłącznie metody jednokrokowe. W niniejszym artykule opisano proces opracowania metody kilkukrokowej będącej uzupełnieniem innych metod. W pierwszej części przedstawiono zalety metody kilkukrokowej. W części teoretycznej skupiono się na dwukrokowej metodzie Adams-Bashforth-Moultonaa (ABM) oraz możliwości odpowiedniego wykorzystania wzorów do zmiennej wielkości kroków (stepsize). Następnie opracowano algorytm służący rozwiązywaniu równań różniczkowych z zastosowaniem wzorów ABM i dostosowaniem stepsize w zależności od odchylenia pomiędzy prognozą a korektą. Opisano to w części Realizacja. W końcowej części artykułu (Výsledky - Wyniki) metodę ABM porównano z metodami RK4 i Hanna.