Diagnosing Newton's Solver Convergence Failures in the Initialization of Modelica Models

Francesco Casella[1] Bernhard Bachmann[2] Karim Abdelhak[2] Philip Hannebohm[2] Teus van der Stelt[3]

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy,
{francesco.casella}@polimi.it
[2]Faculty of Engineering and Mathematics, University of Applied Sciences Bielefeld, Germany,
{bernhard.bachmann,karim.abdelhak,philip.hannebohm}@hsbi.de
[3]Asimptote BV, The Netherlands, teus.vanderstelt@asimptote.com

Abstract

The convergence failure of iterative Newton solvers during the initialization of Modelica models is a serious show-stopper, particularly for inexperienced users. This paper presents the implementation in the OpenModelica tool of methods presented by two of the authors in a previous paper, to help diagnosing and resolving these convergence failure by providing ranked lists of potentially critical start attributes that might need to be fixed in order to successfully achieve convergence. The method also provides library developers with useful information about critical nonlinear equations, that could be replaced by equivalent, less nonlinear ones, or approximated by homotopy for more robust initialization.

Keywords: Initialization failure, Newton's method, Debugging

1 Introduction

The Modelica language (Mattsson, Elmqvist, and Otter 1998; Modelica Association 2023) has firmly established itself as a standard for system-level simulation of cyber-physical systems. The combination of declarative, equation-based modelling style and of modular, object-oriented composition of structured system models allows for flexible, convenient, and self-documented modelling of complex systems. Engineers can easily assemble system models using a drag-and-drop methaphor from libraries of reusable components, whereas domain experts can conveniently develop new component models by describing their behaviour through a-causal differential-algebraic equations. The solution of the overall system equation is entirely off-loaded from the modeller to the simulation tools, which use sophisticated symbolic and numerical methods to generate efficient and numerically robust code to solve the system equations and produce simulation results.

The price to pay for this flexibility and ease of use in building system models is that resulting system models often require the solution of implicit equations, both during initialization and during simulation. In some cases, implicit equations can be automatically solved in closed form using symbolic algorithms; however, in many cases, particularly for systems of non-trivial size, numerical methods must be employed.

The numerical solution of *linear* implicit equations poses no particular challenges, as long as the model is well-posed, so that the system is non-singular; automatic scaling techniques, see, e.g., (Casella and Braun 2017) can be used to improve the condition number of the A matrix of systems that are nearly singular because of bad scaling. Conversely, the numerical solution of *nonlinear* systems of equations requires the use of iterative methods, usually some kind of quasi-Newton algorithm, which can be problematic.

Iterative quasi-Newton methods require a suitable initial guess of the solution to get started; provided that the system Jacobian is non-singular in the sought-after solution, they provide fast convergence to the solution if the initial guess is close enough to it. Good quasi-Newton algorithms implement additional strategies such as damped steps, enforcing min-max limits on variables, etc., to enlarge the set of initial guesses that lead to successful convergence and to cope with poorly chosen initial guess values. Unfortunately, it is a sad fact of life that these algorithm sometimes fail to converge.

If convergence failures happen during *simulation*, where the values obtained at the previous time step (or some suitable extrapolation thereof) are used as initial guess, an effective strategy to cope with the failure is simply to retry with a shorter time step; as long as the variables of the problem are changing continuously, it is possible to get an initial guess which is arbitrarily close to the solution by picking a short enough time step, thus ensuring convergence.

The most critical scenario from a practical standpoint is the convergence failure during *initialization*, where `start` attributes are used as initial guess, because there is no easy backup strategy available in that case. If the initialization problem fails to converge to a meaningful solution, the model at hand is pretty useless.

The problem of convergence failure during initializa-

tion is compounded by the fact that the feedback provided to the modeller by Modelica tools is very low-level, compared to the nice, high-level, modular modeling style of the Modelica language. What one typically gets is an ugly log message from the solver, showing the selected initial guess values and some error message such as "The nonlinear solver failed to converge during initialization, please provide better start attributes". The question is: how is the poor end user supposed to figure them out?

A naïve strategy in this case is to inspect the list of start values that led to the failure, comparing them to one's best estimate of what their value at the solution should be; values which are obviously very far from the estimated solution can be improved one by one. However, this is a very frustrating and time-consuming strategy, particularly in the case of problems with dozens or hundreds of unknowns. For reasons that will become clear in Section 2, in some cases very large errors in the initial guess are not a problem for convergence, whereas comparatively small errors could be critical in other cases, so one may end up wasting a large amount of time fixing `start` attributes which are not critical at all. If the problem at hand has 100 unknowns, is it really necessary to fix all the 100 start attributes, or maybe fixing only a few ones will suffice to attain convergence? Answering to this question is the main goal of the methods discussed in this paper.

More specifically, based on results previously published by Casella and Bachmann (2021), this paper discusses how a Modelica tool can rank the initial guess values of failing nonlinear systems of equations, highlighting the ones which are more likely to be responsible for the convergence failure, thus helping the end user to succeed in solving the initialization problem and getting the simulation to run. It also shows how critical nonlinear equations can be identified, so that they can be, e.g., approximated by means of homotopy by the model developer. The proposed methods were implemented in the OpenModelica tool (Fritzson et al. 2020) and successfully tested against the test cases discussed in that paper.

The paper is structured as follows: Section 2 reviews the methods proposed by Casella and Bachmann (2021), introducing the notation and the main results; Section 3 discusses the implementation in the OpenModelica tool, while Section 4 shows some of the results obtained on some of the test cases discussed in (Casella and Bachmann 2021). Section 5 concludes the paper with final remarks and suggestions for future work.

2 Method

This section introduces the notation for the problem at hand and recalls the main results of (Casella and Bachmann 2021) which are relevant for the discussion.

2.1 Notation

Consider the problem

$$f(x) = 0, \tag{1}$$

where $x \in \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}^m$ is a vector function which is continuously differentiable in an open neighbourhood $\mathfrak{D}$ of the solution $\bar{x}$, $f(\bar{x}) = 0$. In the context of this paper, problem (1) corresponds to a strong component of the initialization problem of a Modelica model, where x is a subset of the set of unknowns of such problem, namely state variables, their derivatives, algebraic variables, and unknown parameters with `fixed = false` attribute. In case homotopy-based initialization is used (Sielemann et al. 2011), the considered problem is the one at $\lambda = 0$, for which the intial guess is taken from the `start` attributes.

Denote the Jacobian matrix of function $f(x)$ with respect to x as $f_x(x)$. Assume the vector of the unknowns x is suitably ordered, so that it can be split into two subvectors $w \in \mathbb{R}^q$ and $z \in \mathbb{R}^{m-q}$

$$x = \begin{bmatrix} w \\ z \end{bmatrix}, \tag{2}$$

w being the smallest possible sub-set of x such that

$$f_x(x) = J(w), \tag{3}$$

i.e., the Jacobian matrix of $f(x)$ depends only on w and not on z. Thus, w are defined as the *nonlinear variables* of the problem, while z are defined as the *linear variables* of the problem.

Assume the equations in (1) are ordered so that $f(x)$ can be split into two vector functions $n(x)$ and $l(x)$, $n : \mathbb{R}^m \to \mathbb{R}^p$, $l : \mathbb{R}^m \to \mathbb{R}^{m-p}$

$$f(x) = \begin{bmatrix} n(x) \\ l(x) \end{bmatrix}, \tag{4}$$

where $n(x)$ contains the non-linear equation residuals and $l(x)$ contains the linear equation residuals. System (1) is thus split into *nonlinear equations* $n(x) = 0$ and *linear equations* $l(x) = 0$. The system (1) can then be rewritten as

$$f\left(\begin{bmatrix} w \\ z \end{bmatrix}\right) = g(w) + f_z z. \tag{5}$$

where $g(w)$ contains all the nonlinear terms of the system equations, while $f_z z$ denotes the linear part of the system equations.

The solution $\bar{x}$ can be computed iteratively by Newton-Raphson's method, which requires to solve the following linear equation at each iteration j

$$f_x(x_{j-1})(x_j - x_{j-1}) = -f(x_{j-1}), \qquad j = 1, 2, \cdots \tag{6}$$

starting from a given initial guess x_0, which is obtained from the `start` attributes of the unknowns x.

One very important remark is due when tearing is used to solve the implicit nonlinear system. In this case, the set of unknowns x is the set of *tearing* (or iteration) variables, whereas the set of equations (1) refers to the set of *residual* equations. In this case, determining which are the nonlinear variables and equations of the torn system requires to

DOI
10.3384/ecp218109

consider the dependencies between each tearing variable and each residual equation introduced by the torn equations, which may themselves be linear or nonlinear. For example, the system

$$\sin(a) + b = 1 \tag{7}$$
$$a + b + \cos(c) = 2 \tag{8}$$
$$a + b + c = 0 \tag{9}$$

can be solved by selecting a and c as tearing variables:

$$b := 1 - \sin(a) \tag{10}$$
$$a + b + \cos(c) - 2 = 0 \tag{11}$$
$$a + b + c = 0; \tag{12}$$

where the unknown vector of the torn system is $x = [a \quad c]'$. Although the last of the two residual equation looks linear at first glance, it actually depends non-linearly on a through the nonlinear torn equation. Therefore, both unknowns and both residual equations of the torn system are actually non-linear in this case.

2.2 Theoretical Background

This section recalls the main theoretical results of Casella and Bachmann (2021), putting them in the specific context of a Modelica tool solving a strong component of the initialization problem.

Theorem 1. ***(Superlinear convergence close to the solution)****. If the Jacobian $f_x(\bar{x})$ is non-singular in the solution $\bar{x}$ and Lipschitz-continuous in a neighbourhood of $\bar{x}$, for all x_0 sufficiently close to x, the sequence $\{x_j\}$ of the solutions of* (6) *converges not less than quadratically to $\bar{x}$.*

Theorem 2. ***(Convergence in the linear case)****. If Equation* (1) *is linear and f_x is non-singular, then Netwon's algorithm converges in one step, irrespective of the chosen initial guess x_0.*

Theorem 3. ***(Convergence in the mixed linear-nonlinear case)****. If Newton's algorithm is initialized with a first guess*

$$x_0 = \begin{bmatrix} w_0 \\ z_0 \end{bmatrix}, \tag{13}$$

the values of the approximated solution x_j at each step $j > 0$ only depend on the guess values of the nonlinear variables w_0, regardless of the choice of guess values of the linear variables z_0.

Theorem 4. ***(Linear residuals after the first iteration)****. The residuals of the linear equations in system* (1) *after the first iteration of Newton's algorithm are zero, i.e. $l(x_1) = 0$, regardless of the initial guess values x_0.*

Theorems 1 and 2 are well-known, whereas Theorems 3 and 4 were first stated by Casella and Bachmann (2021) and have two very important implications for the problem at hand:

1. only *nonlinear* equations $n(x) = 0$ and only initial guesses of *nonlinear* variables w_0 matter for convergence;

2. there is no reason to bother about `start` attributes of *linear* variables

For the subsequent discussion, it is necessary to answer the following question: how can we determine if the initial guess x_0 is close enough to the solution $\bar{x}$, so we can expect that convergence will be achieved? One naïve answer could be to consider this criterion:

$$\left\| f(x_1) \right\| \ll \left\| f(x_0) \right\|, \tag{14}$$

i.e., if the residual after the first iteration is much smaller than the residual computed with the initial guess, then it's likely that the residual after the next iteration will be even smaller and convergence will be achieved soon.

In fact, due to the consequences of Theorems 3 and 4, this is not a sound criterion: Theorem 3 suggests to trivially set $z_0 = 0$, which may lead to large values of the linear residuals $l(x_0)$, while Theorem 4 states that after one iteration, $l(x_1) = 0$ no matter what. Hence, the norm of the overall residual vector may dramatically decrease after one iteration, just because large linear residuals becoming zero in one shot; this is by no means an indication that the nonlinear variables are close to convergence.

This observation leads to the need of defining a better criterion for being close to convergence.

Definition 1. *With reference to Newton's iteration* (6)*, define the nonlinear residual at iteration point x_{k-1} as*

$$r(x_{k-1}) = f(x_{k-1}) + f_z(z_k - z_{k-1}). \tag{15}$$

This definition basically removes all the linear components that will vanish after one iteration from the residual at step $k-1$. Another possible interpretation, with reference to the decomposition shown in (5), is that it computes the residual of the nonlinear part of the system equations only, which is the only one that is relevant for possible convergence issues.

We can then replace the right hand side of the naïve criterion (14) with the nonlinear residual, thus leading to this heuristic criterion for being close to convergence:

$$\left\| f(x_1) \right\| \ll \left\| r(x_0) \right\|. \tag{16}$$

This criterion is heuristic in the sense that it is possible to build on purpose counter-examples in which this condition holds, but yet Newton's method does not converge to a solution. However, in most practical cases, if the residual $f(x_1)$ after the first iteration is much smaller than the nonlinear residual computed with the initial guess $r(x_0)$, then the residual $f(x_2)$ after the second iteration will be even smaller, and the method will converge to a solution in a few iterations.

The next step is to come up with the formulation of some indicators that can be used to express relationship (16) in an equivalent way. This requires to introduce the following definitions.

Definition 2. *Consider Newton's iteration* (6)*. Assume that the function $f(x)$ is three times continuously differentiable in an open neighbourhood $\mathfrak{D}$ containing the initial guess x_0 and the result of the first iteration x_1. Denote the i-th component of function $f(x)$ as $f^i(x)$, its Jacobian matrix with respect to x as $f^i_x(x)$, and its Hessian matrix as $f^i_{xx}(x)$. One can write the following Taylor expansion*

$$f^i(x_1) = f^i(x_0) + f^i_x(x_0)(x_1 - x_0) + \\ + \frac{1}{2}(x_1 - x_0)' f^i_{xx}(x_0)(x_1 - x_0) + h^i(x_1, x_0) \quad (17)$$

which implicitly defines the higher-order residual functions $h^i(\cdot,\cdot)$.

Definition 3. *Define the coefficients $\alpha_i > 0$, $i = 1, \cdots, m$, such that*

$$|h^i(x_1, x_0)| = \alpha_i \left\| r(x_0) \right\|_\infty, \quad (18)$$

i.e.,

$$\alpha_i = \frac{\left| f^i(x_1) - \frac{1}{2}(w_1 - w_0)' f^i_{ww}(w_0)(w_1 - w_0) \right|}{\left\| r(x_0) \right\|_\infty} \quad (19)$$

and let

$$\alpha = max(\alpha_i). \quad (20)$$

Remark 1. *It may happen that when computing $f(x_1)$ some of the residuals cannot be computed because x_1 is out of their range of definition, e.g. because of logarithms or square roots getting a negative argument. In this case, it is necessary to perform a* damped *Newton iteration:*

$$f_x(x_{j-1})(x^*_j - x_{j-1}) = -\lambda f(x_{j-1}), \qquad j = 1, 2, \cdots \quad (21)$$

By taking a small enough positive value of λ, it is possible to get x_1 arbitrarily close to x_0, thus avoiding validity range issues. The Taylor expansion can be used to define $h^i(x^*_1, x_0)$ leading to the following modified definition of α_i

$$|h^i(x^*_1, x_0)| = \alpha_i \lambda^3 \left\| r(x_0) \right\|_\infty, \quad (22)$$

motivated by the fact that $h^i(x^*_1, x_0)$ is a third-order term in the Taylor expansion, so it is expected to shrink as λ^3, thus making the definition asymptotically invariant as $\lambda \to 0$. Hence, α_i can be computed as

$$\alpha_i = \frac{\left| f^i(x^*_1) - (1-\lambda) f(x_0) - \frac{1}{2}(w^*_1 - w_0)' f^i_{ww}(w_0)(w^*_1 - w_0) \right|}{\lambda^3 \left\| r(w_0) \right\|_\infty} \quad (23)$$

Definition 4. *Define the* curvature factor Γ_{ijk} *of the i-th nonlinear equation with respect to variables w_j, w_k after the first iteration as*

$$\Gamma_{ijk} = \left| \frac{1}{2} \frac{\partial^2 g^i(w_0)}{\partial w_j \partial w_k} \frac{(w_{1,k} - w_{0,k})(w_{1,j} - w_{0,j})}{\left\| r(x_0) \right\|_\infty} \right| \quad (24)$$
$$i = 1, \ldots, p,$$
$$j = 1, \ldots, q,$$
$$k = 1, \ldots, q.$$

Equipped with these definitions, we can now state the following Theorem:

Theorem 5. *Given a constant $\beta > 0$,*

$$\sum_{jk} \Gamma_{ijk} \leq \beta \qquad \forall i = 1, \cdots, p. \quad (25)$$

implies that

$$\left\| f(x_1) \right\|_\infty \leq (\alpha + \beta) \left\| r(x_0) \right\|_\infty \quad (26)$$

The main consequence of Theorem 5 is that if $\alpha_i \ll 1$ and $\Gamma_{ijk} \ll 1 \; \forall i, j, k$, then $\alpha \ll 1$ and it will be possible to pick a value of $\beta \ll 1$ that satisifes (25); hence, $\alpha + \beta \ll 1$ and therefore, thanks to (26), the heuristic convergence condition (16) will be satisfied. In short, if all the α_j and Γ_{ijk} coefficients are small, Newton's method is very likely to converge in a few steps. This condition is sufficient but by no means necessary to achieve convergence, in particular if more robust quasi-Newton methods are used to actually solve (1). However, one can put forth the following heuristic argument:

> *If the (quasi-)Newton solver fails to converge, and if some of the α_i and Γ_{ijk} coefficients are much larger than one, they are likely to be pointing out the initial guesses and nonlinear equations that are responsible for the convergence failure.*

Specifically, large α_i values will indicate that the i-th equation is involved, whereas large values of Γ_{ijk} will indicate that the i-th equation and the initial guesses of w_j and w_k are involved.

Note, from definition (24), that the Γ_{ijk} indicators exploit second-order information provided by the system Hessian, by weighting the magnitude of the change of the solution estimate $(w_{1,j} - w_{0,j})$ after the first Newton step with the curvature of the residual function.

This is a crucial point: the fact that the initial guess value of some variables are very far from the solution, so that $(x_{1,k} - x_{0,k})$ is very large, is *per se* not a relevant indicator of potential convergence issues; in fact, as Theorems 3 and 4 clearly imply, it is not relevant at all for linear variables z. It becomes more and more relevant as the residual depends on the variable in a strongly nonlinear way, as measured by the value of the second derivative.

DOI 10.3384/ecp218109

Regarding the α_i indicators, it is clear from their definition in (17)-(18) that they instead account for the combined effect of terms of order higher than two on the convergence to the solution. Unfortunately, these indicators only point to the involved nonlinear equations, but they cannot discriminate which of the initial guess(es) of nonlinear variables $w_{0,k}$ showing up in those equations are to blame for the convergence issues.

Additional indications about initial guess values potentially being responsible for convergence failure can be obtained by exploiting the following Theorem:

Theorem 6. *The sensitivity of the solution x_1 after the first NR iteration, with respect to changes in the initial guess x_0, can be computed as:*

$$\frac{\partial x_1}{\partial x_0} = \Sigma, \tag{27}$$

where

$$H_i = (w_1 - w_0)' f^i_{ww}(w_0) \tag{28}$$

$$H = \begin{bmatrix} H_1 \\ H_2 \\ \cdots \\ H_p \end{bmatrix} \tag{29}$$

$$\Sigma = -\left[f_x(w_0)\right]^{-1} \begin{bmatrix} H_{p\times q} & 0_{p\times(m-q)} \\ 0_{(m-p)\times q} & 0_{(m-p)\times(m-q)} \end{bmatrix}. \tag{30}$$

When computing the sensitivity Σ at the solution $x_0 = \bar{x}$, since $w_1 = w_0 = \bar{w}$, it follows that $H = 0$, so that $\Sigma = 0$. This means that if an initial guess equal to the solution plus an infinitesimally small perturbation $x_0 = \bar{x} + \delta x$ is chosen, the solution x_1 after the first NR iteration is not affected at all. This is consistent with the fact that $f(x)$ can be approximated as a linear function in a small neighbourhood of the solution $\bar{x}$, so that Theorem 2 guarantees that the first NR iteration converges to the solution $\bar{x}$ in just one iteration, irrespective of the initial guess.

If the initial guess x_0 is close enough to the solution $\bar{x}$ that the function $f(x)$ is still approximately linear in a neighbourhood containing x_0 and $\bar{x}$, then the same behaviour is preserved, i.e., the result after the first iteration will be insensitive to small changes of the initial guess, so $\Sigma \approx 0$.

As x_0 is chosen farther away from the solution $\bar{x}$, nonlinear effects kick in, accounted for by matrix Σ, which can be then considered an indicator of how far the initial guess is from the sweet spot of NR convergence. In particular, $\left|\sigma_{jj}\right| \ll 1$ means that the effect on $w_{1,j}$ of applying a certain perturbation to $w_{0,j}$ will be much less than the perturbation itself, meaning that the nonlinear effects are moderate, whereas if $\left|\sigma_{jj}\right| \gg 1$ the effect of such perturbation will be amplified after the first iteration, which is not consistent with the behaviour close to convergence. Hence, it is possible to put forth an argument similar to the one concerning α_i and Γ_{ijk} indicators:

> *If the Newton solver fails to converge, and if $\left|\sigma_{jj}\right| > 1$ for some j, then the initial guesses $w_{0,j}$ are likely to be responsible for the convergence failure.*

Similarly to the Γ_{ijk} indicators, the σ_{jj} indicators rely on second-order information provided by the system Hessian to assess the effect of the nonlinearity of the system of equations on the convergence of Newton's method. Contrary to Γ_{ijk}, σ_{jj} indicators only provide information about potentially problematic initial guesses $w_{0,j}$, while not providing any information about the involved nonlinear equations.

An interesting property of σ_{jj} coefficients, when computed from sets of equations containing quantities with physical units, is that they are dimensionless, so they are invariant with respect to the choice of units of the problem, and in general to the scaling of the system of equations. This makes them readily computable from the Modelica equations in SI units, even if the equations and variables are badly scaled from a numerical point of view.

The α_i and Γ_{ijk} indicators are also invariant with respect to the choice of the units of the unknowns x; however, they are *not* invariant with respect to the choice of the units of the *residual equations* $f^i(x)$. In fact, the definitions of both indicators rely on the infinity norm of the residual $\left\|r(x_0)\right\|_\infty = \max_i \left|r^i(x_0)\right|$, which is conceptually ill-defined if the residuals $r^i(x_0)$ have non-homogenous units: what is the maximum between 70 kg/s and 10^8 W in a system of equation with a steam flow rate equation and a power balance equation?

Hence, a proper implementation of the computation of α_i and Γ_{ijk} requires to employ *normalized* function residuals, which can be computed as explained in (Casella and Braun 2017). Once the residuals are nomalized, the α_i and Γ_{ijk} indicators will be invariant to the choice of units of the problem and to its scaling in general.

2.3 Diagnosing Newton Solver Failures

In case the quasi-Newton method of choice fails to solve one of the nonlinear strong components of the initialization problem (in the case $\lambda = 0$ if homotopy is used), it is possible to compute the α_i, Γ_{ijk} and σ_{jj} indicators as explained in the previous subsection. They can then be ranked in descending order and displayed.

The initial guesses $w_{0,j}$ and $w_{0,k}$ corresponding to the largest values of Γ_{ijk} and $\left|\sigma_{jj}\right|$, as well as the unknowns showing up in the equations with the larger values of α_i, are the most likely to be responsible for the solver failures, so their start attribute should be checked and possibly fixed first. If it is recognized that they have a different start value than what was expected, e.g. 300 K for a flame temperature instead of 1800 K, this can be easily pin-pointed and corrected. Otherwise, they could be tentatively increased or decreased based on the sign of $(w_{1,j} - w_{0,k})$.

This method will be particularly helpful in the case of large systems with many unknowns and only a few critical

start attributes, that will be spotted immediately at the top of the list of ranked indicators.

This method can also help library developers to pinpoint equations that are heavily involved in the failure because of their strong nonlinearity, as indicated by large α_i and Γ_{ijk} values. Such equations could be replaced with other equivalent ones that are less nonlinear, e.g. $a = b * c$ instead of $a/b = c$, or replaced by less nonlinear approximated equations by means of homotopy.

3 Implementation

The method illustrated in the previous Section was implemented in the OpenModelica runtime, where it can be activated by setting the simulation flag `-lv=LOG_NLS_NEWTON_DIAGNOSTICS`. Jacobians are obtained using the same functions that are employed to get them for the quasi-Newton solvers used for simulation, which can either use code obtained with symbolic differentiation or resort to numerical finite differences when that is not possible. Hessians are computed numerically from Jacobians by means of finite differences.

Although it would be conceptually possible to use symbolic differentiation also for the Hessian computation, developing the code to do so was not really worth the effort: the second derivatives are used to generate heuristic indicators, so a high precision is not really required, so that the trivial task of computing them by numerical differentiation was chosen instead. Computational times are also not an issue, since all the required computations are only carried out once and normally take much less than a second also for systems with hundreds of unknowns, so simplicity of implementation was favoured over efficiency. This also lead to the choice of always using the LAPACK dense solver DGESV to solve the linear equations of the Newton step, also in case of systems with high sparsity ratio.

As mentioned in Section 2.1, the classification of linear and nonlinear equations and variables when tearing is used must also consider the nonlinearities introduced by the torn equation. This information could have been inferred symbolically by extending the structural analysis algorithm of the OpenModelica backend; however, this would have taken a significant development effort. It was therefore chosen to follow a simpler numerical approach instead.

Linear equations $l(x) = 0$ are therefore identified by checking if their residual after the first Newton step $f^i(x_1) \approx 0$, see Theorem 4. In case a damped Newton step needs to be computed to avoid out-of-range residual evaluations, the following condition is checked:

$$f^i(x_1^*) + (\lambda - 1) f^i(x_0) \approx 0 \tag{31}$$

Linear variables z are instead identified by checking if the corresponding column of the Hessians is close to zero.

4 Test Results

At the time of this writing, the implementation of the method in release 1.25.0 of OpenModelica is nearly complete, except for two features:

- getting numerical Jacobians when symbolically differentiated Jacobians are not available;
- scaling the residuals when computing the nonlinear residual norm $\left\| r(x_0) \right\|_\infty$

This unfortunately still prevents from testing the implemented method on the initialization problems of physical models using SI units. In particular, the steady-state initialization of thermo-hydraulic problems is notoriously prone to convergence failures, so it could benefit immensely from this type of diagnostics, but unfortunately it involves badly scaled variables such as pressures in Pa, as well as medium models such as the IF97 water/steam model, that cannot be symbolically differentiated.

The implementation was therefore tested on the three example cases shown in (Casella and Bachmann 2021), namely a simple thermo-hydraulic system, a nonlinear DC circuit, and an AC circuit. In all the three cases, the exact analytic solution of the system is known, so it is possible to set the start attribute of the unknowns at various distances from the solution, testing the ability of the proposed method to highlight the ones that need to be fixed in order to ensure the convergence of the Newton method. These test problems use well-scaled units with all the unknowns having order of magnitude of unity, so there is no need to scale the elements of $\left\| r(x_0) \right\|_\infty$ to obtain good results.

The computed values of the α_i, Γ_{ijk}, and $\left|\sigma_{jj}\right|$ indicators were very close to the ones that are reported in (Casella and Bachmann 2021), which were computed by ad-hoc, manually written Matlab code, thus validating the C-code implementation in the OpenModelica runtime. In order to obtain the same results, tearing had to be de-activated, so that the raw nonlinear system written in Modelica was passed directly to the diagnostic routine.

The results of one test case are shown here. The test system is the model of an electrical DC circuit, where the series connection of N resistors and one diode is connected to an ideal voltage source, which provides a certain fixed power P. The system is described by the following set of implicit equations:

$$i - i_s \left(e^{v_d / v_t} - 1 \right) = 0 \tag{32}$$

$$vi - P = 0 \tag{33}$$

$$v - \sum_{j=1}^{N} v_j - v_d = 0 \tag{34}$$

$$v_j - Ri = 0, \qquad j = 1 \cdots N \tag{35}$$

where i_s, v_t, P, R are known parameters, $x = \begin{bmatrix} i & v_d & v & v_1 & v_2 & \cdots & v_N \end{bmatrix}'$, $w = \begin{bmatrix} i & v_d & v \end{bmatrix}'$, $z = \begin{bmatrix} v_1 & v_2 & \cdots & v_N \end{bmatrix}'$.

DOI 10.3384/ecp218109

Note that the system has $N+1$ linear equations and only two nonlinear equations: the first is very strongly nonlinear, due to the exponential term in the diode equation, while the second, containing the voltage-current product, is only mildly nonlinear. The system has three nonlinear variables, which are the only ones for which the initial guess value matters, according to Theorem 3.

Taking $i_s = 6.9144 \cdot 10^{-13}$, $v_t = 25 \cdot 10^{-3}$, $P = 10.7$, $R = 1$, $N = 10$, the system has an exact solution $i = 1$, $v_d = 0.7$, $v = 10.7$, $v_j = 1$. If the start attributes of i, v_d, and v are reduced by 10% with respect to the known solution, the quasi-Newton solvers of the OpenModelica runtime fail to solve the system.

When activating the Newton Diagnostics runtime flag, OpenModelica generates the diagnostic output shown in Fig. 1.

The first table shows results by variable, ranked in descending order of potential criticality; for each variable, the largest value of Γ_{ijk} and $\left|\sigma_{jj}\right|$ that involve it are reported. The result unequivocally points out some very serious problem with the start attribute of v_d; indeed, due to the exponential term in the diode equation and to the small value of the v_t parameter, an error of 10% in the initial guess for v_d is enough to cause convergence failure. Conversely, the same 10% error in the start values of i and v, which are involved in the mildly nonlinear equation $vi - P = 0$, is not problematic at all. In fact, increasing the start attribute of v_d from 0.63 to 0.66 is enough to ensure the convergence of the quasi-Newton algorithm used by OpenModelica to solve the nonlinear system, without any need to touch the start attributes of i and v.

The second table shows results by equation, ranked in descending order of potential criticality. The "Eq no." index refers to the ordering of the residuals in the runtime solver, in case one wanted to inspect it in detail, whereas the "Eq idx" index refers to the index of the equations in the equation-based debugger (Sjölund et al. 2014), which displays the solved equations after structural analysis and symbolic manipulation, see Fig. 2. The diode equation is clearly indicated as the one creating trouble, while, the power equation is not creating any significant problem with the selected start attributes for its variables i and v.

For more details about the other test cases, the interested reader is referred to (Casella and Bachmann 2021).

5 Conclusions

The convergence failure of Newton's method during the solution of nonlinear systems of equations for the initialization of Modelica model is a possible critical outcome of the attempt of simulating Modelica models with implicit nonlinear equations, particularly when initializing models in steady state. Such an occurrence is a serious issue for Modelica tool users and can be a show-stopper for inexperienced users that have no idea how to fix the problem.

This paper presents the implementation in the OpenModelica tool of methods originally devised by Casella and Bachmann (2021) to diagnose such failures and help resolving them in a user-friendly manner. The presented implementation outputs lists of potentially problematic start attributes and nonlinear equations, ranked in descending criticality order.

In the authors' opinion, the presented methods provide much needed support in the debugging of failing Modelica models, particularly if coupled with user-friendly equation-based debuggers such as the one presented in (Sjölund et al. 2014).

The implementation was successfully tested in some illustrative test cases taken from Casella and Bachmann (2021). Unfortunately, at the time of this writing it was not yet possible to compute the normalized function residuals, as explained at the end of Section 2.2, because that requires a substantial refactoring of the OpenModelica runtime code. This prevented testing the method on more complex and realistic Modelica models using SI unit variables, whose residuals are usually quite badly scaled. Such a refactoring is currently on-going and planned to be finalized for the next 1.26.0 release of the software.

Future work will also involve a tighter integration of these run-time methods with the equation-based debugger of OpenModelica, which currently only provides static information elaborated at compile time.

References

Casella, Francesco and Bernhard Bachmann (2021). "On the choice of initial guesses for the Newton-Raphson algorithm". In: *Applied Mathematics and Computation* 398.125991, pp. 1–18. DOI: 10.1016/j.amc.2021.125991.

Fritzson, Peter et al. (2020). "The OpenModelica integrated environment for modeling, simulation, and model-based development". In: *Modelling, Identification, and Control* 4, pp. 241–285. DOI: 10.4173/mic.2020.4.1.

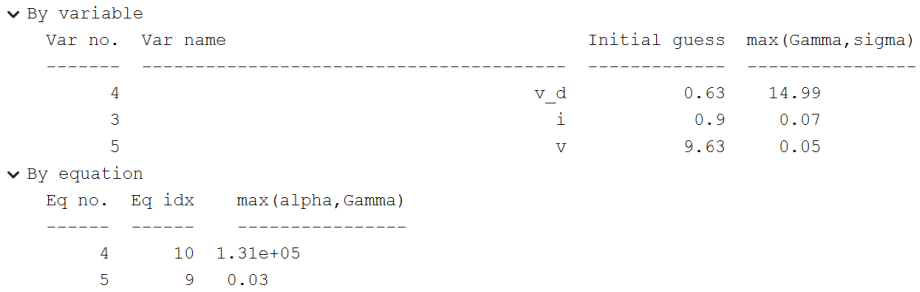
Casella, Francesco and Willi Braun (2017-12). "On the importance of scaling in equation-based modelling". In: *8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, EOOLT 2017*. Wessling, Germany, pp. 3–7. DOI: 10.1145/3158191.3158192.

Sielemann, Michael et al. (2011-03). "Robust Initialization of Differential-Algebraic Equations Using Homotopy". In: *Proceedings 8th International Modelica Conference*. Ed. by C. Clauss. Modelica Association. Dresden, Germany, pp. 75–85. ISBN: 978-91-7393-096-3. DOI: 10.3384/ecp1106375. URL: http://www.ep.liu.se/ecp/063/010/ecp11063010.pdf.

Sjölund, Martin et al. (2014-03). "Integrated Debugging of Equation-Based Models". In: *Proceedings 10th International Modelica Conference*. The Modelica Association. Lund, Sweden, pp. 195–204. ISBN: 978-91-7519-380-9. DOI: 10.3384/ECP14096195.

Mattsson, S. E., H. Elmqvist, and M. Otter (1998). "Physical system modeling with Modelica". In: *Control Engineering Practice* 6.4, pp. 501–510.

Modelica Association (2023-02). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/maint/3.6/MLS.html.



By variable

Var no.	Var name	Initial guess	max(Gamma,sigma)
4	v_d	0.63	14.99
3	i	0.9	0.07
5	v	9.63	0.05

By equation

Eq no.	Eq idx	max(alpha,Gamma)
4	10	1.31e+05
5	9	0.03

Figure 1. OpenModelica diagnostic output on the DC circuit example: initial guess values and equations most critical for convergence are shown, ranked in descending order of criticality (rightmost column).

28	initial	non-linear, unknowns: 13, iteration variables: 13
1	initial	(residual) v_[9] - R * i = 0
2	initial	(residual) v_[8] - R * i = 0
3	initial	(residual) v_[7] - R * i = 0
4	initial	(residual) v_[6] - R * i = 0
5	initial	(residual) v_[5] - R * i = 0
6	initial	(residual) v_[4] - R * i = 0
7	initial	(residual) v_[2] - R * i = 0
8	initial	(residual) v_[1] - R * i = 0
9	initial	(residual) v * i - P = 0
10	initial	(residual) i - i_s * (-1.0 + exp(v_d / v_t)) = 0
11	initial	(residual) v_[3] - R * i = 0
12	initial	(residual) (-v_[1]) - v_[2] - v_[3] - v_[4] ...v_[7] - v_[8] - v_[9] - v_d - v_[10] + v = 0
13	initial	(residual) v_[10] - R * i = 0

Figure 2. OpenModelica declarative debugger output on the DC circuit, each equation index (Eq idx) is shown in the first column.

DOI
10.3384/ecp218109