

DAWRS: A Differential – Algebraic System Solver by the Waveform Relaxation Method

Argimiro R. Secchi Manfred Morari*

Chemical Engineering 210-41
California Institute of Technology
Pasadena, CA, 91125, USA

Evaristo C. Biscaia Jr.

Engenharia Química - Sala G115
COPPE - Univers. Federal do Rio de Janeiro
Rio de Janeiro, RJ, 21945, Brazil

Abstract

We investigate the concurrent solution of low-index differential-algebraic equations (DAE's) by the waveform relaxation (WR) method, an iterative method for system integration. We present our new simulation code, DAWRS (Differential – Algebraic – Waveform Relaxation Solver), to solve DAE's on parallel machines using the WR methods, and describe new techniques to improve the convergence of such methods. As experimental results, we demonstrate the achievable concurrent performance to solve DAE's for a class of applications in chemical engineering.

1 Introduction

There are two basic concurrent simulation paradigms for solving DAE's: first, "direct methods" exploit the parallelism across the existing sequential algorithms. Such methods maintain all numerical characteristics of the original algorithms, and their performance depends on how well the parallelism is exploited, see [1] for more detail. Second, "dynamic iterative methods," or WR methods, exploit the parallelism across the system, iterating independent solutions of different parts of the overall system. The performance for WR methods depends, mainly, on their convergence and scheduling characteristics.

Here, we investigate the solution of DAE's by the WR method implemented in our DAWRS package, an application-independent C-based concurrent DAE solver. We utilize distillation column networks to demonstrate the achievable performance of DAWRS on multicomputers such as Symult s2010 and Intel iPSC860. Such problems are modeled by large-scale, sparse, and nonsymmetric DAE's, with a natural imbalance and unequal activity (latency) in their residuals, which can be well exploited by the WR methods.

We present new techniques to improve both the local and global convergence of the WR methods. We show significant gains in performance as a result of our new approaches to manipulation of the waveforms. Finally, we discuss the system partitioning steps in which the DAE's to be solved must be partitioned into several lower-order subsystems.

*Correspondence should be sent to: phone (818)356-4186, fax (818)568-8743, e-mail mm%imc@iago.caltech.edu

2 Integration Layer

As the WR method exploits the parallelism across the system, each subsystem may use its own integrator according to its needs (explicit, implicit, ...). Here, we consider only the case where all subsystems use the same integrator. We use DASSL [2] to carry out the subsystem integration, because of its efficiency to solve low-index systems of DAE's of the form

$$\begin{aligned} F(t, y, \dot{y}, u) &= 0 \\ y(t_0) &= y_0, \quad \dot{y}(t_0) = \dot{y}_0 \end{aligned} \quad (1)$$

where $F: R \times R^N \times R^N \times R^r \rightarrow R^N$ is a nonlinear function, $y(t) \in R^N$ is the vector of state variables, and $u(t) \in R^r$ is the input vector. ODE's are included in this formulation as a special case of DAE's.

DASSL uses a variable stepsize and order, implicit BDF scheme. The variable stepsize is an essential property to exploit the latency of the subsystems. In its stepsize and order selection, the order is lowered or raised depending on if the leading error term in the remainder of the Taylor series expansion form an increasing or decreasing sequence. The new stepsize is chosen so that the error at the new order satisfies

$$M \|y_{n+1} - y_{n+1}^{(0)}\| \leq 1.0 \quad (2)$$

where M bounds the error estimate taking into account the interpolation and local truncation error, and $\|y_{n+1} - y_{n+1}^{(0)}\|$ is the norm of the predictor-corrector difference¹. To control the local error the stepsize is rejected whenever the condition (2) is not satisfied.

3 Algorithm Description

To formulate the decoupled system for the WR method, we rewrite (1), without loss of generality, as

$$\begin{aligned} F_i(t, y_i, \dot{y}_i, d_i, u) &= 0 \\ y_i(t_0) &= y_{i0}, \quad \dot{y}_i(t_0) = \dot{y}_{i0} \end{aligned} \quad (3)$$

where, for $i = 1, 2, \dots, p$, $F_i: R \times R^{p_i} \times R^{p_i} \times R^{2N-2p_i} \times R^r \rightarrow R^{p_i}$, and d_i is the *decoupling vector* which contains the variables from y which are not in y_i and derivatives from \dot{y} not in \dot{y}_i . Now, if we consider d_i

¹ $\|y\|^2 = \frac{1}{N} \sum_{i=1}^N \left(\frac{y_i}{wt_i}\right)^2$, where wt is the weight vector reflecting the relative and absolute error tolerances.

in (3) as an input vector, we can solve (1) by solving iteratively p independent subsystems.

In the following, we represent waveforms by the ordered set of timepoints

$$z_i(\tau) = \{y_i(t) / t \in \tau = [t_a, t_b] \subseteq \tau_f = [t_0, t_f]\} \quad (4)$$

with $z(\tau) = (z_1, z_2, \dots, z_p)^T$ a vector of waveforms, where $[t_0, t_f]$ is the time interval of interest, and

$$v_i(\tau) = \{d_i(t) / t \in \tau \subseteq \tau_f\} \quad (5)$$

as *neighbor waveforms*, with $v(\tau) = (v_1, v_2, \dots, v_p)^T$ held fixed during the integration process for a specific iteration. Thus, the basic structure of a WR algorithm can be abstractly described for a given subinterval $\tau_q = [t_q, t_{q+1}]$ as shown in Figure 1. The convergence is achieved when $\|z^j - z^{j-1}\|_\infty$ is sufficiently small.

```

set  $j = 0$  and  $v^0(\tau_q) = v_0$  (an initial guess)
do
  for  $i = 1, 2, \dots, p$ 
    solve  $\begin{cases} F_i(\tau_q, z_i^j, \dot{z}_i^j, v_i^j, u) = 0 \\ y_i^j(t_q) = y_i(t_q), \dot{y}_i^j(t_q) = \dot{y}_i(t_q) \end{cases}$ 
    set  $j = j + 1$  and  $v^j(\tau_q) = v_{updated}^{j-1}(\tau_q)$ 
until convergence

```

Figure 1: *Basic waveform relaxation algorithm*

It is clear that the WR algorithm consists of three parts: the system partition phase, the integration phase (Section 2), and the relaxation phase. In the system partition phase, DAWRS executes six multi-option well-defined steps (assignment, grouping, ordering, placement, process generation, and neighborhood), each of which can be either totally or partially user-defined. Also, DAWRS allows arbitrary sets of equations to be addressed (named) by implicit mapping, whereas in [3, 4] the approach is device- or template-oriented.

In the assignment process, each unknown variable is associated with an equation of the system of DAE's in which it is involved, but each equation has to hold the correct state variable to maintain the problem consistency [5] (*consistent assignment*). However, a system of DAE's can have several consistent assignments giving different convergence characteristics to the partitioned system. Thus, a combinatorial optimization [5] allied with the information in the *iteration matrix*² is carried out to find an optimal assignment.

After the system assignment, we can group tightly coupled states, ideally yielding more loosely coupled subsystems. The main reason for this grouping step is to get higher convergence rates, [6]. In other words, we want to reduce the contraction constant γ of the contractive map $\|z^j - z^*\|_\infty \leq \gamma \|z^0 - z^*\|_\infty$ to the solution z^* in the waveform space. We utilize the depth-first search [7] and the iteration matrix to give us necessary information about state coupling. Another way to improve the convergence rate is by exploiting the

directionality of information flow within the system by means of iterations like Gauss-Seidel. These methods require some ordering of the system (e.g., coloring) to get reasonable contraction constants.

Usually, the subsystems have different residual evaluation time and different stiffness in the integration process. Thus we need to place them optimally among the processors in order to obtain good load balancing. Also, we may estimate the communication cost to exchange waveforms as another parameter to proceed with the placement step. At this stage we have all information about disconnected subsystems of DAE's; in the next process generation step, we actually create the processes with all clustered subsystems to be loaded into the available processors. Finally, the neighborhood step finds all interconnected subsystems either by means of user information or by perturbation over the state variables, and establishes the pattern of waveform communications during the simulation.

In the relaxation phase DAWRS iterates the waveforms until convergence, according to the selected iterative scheme (Jacobi, asynchronous iterations, etc.). An efficient scheduler process, with a low sequential fraction, controls the system convergence. Also, as successfully implemented in the CONCISE simulator [3], we incorporate a dynamic waveform *splitting* strategy where we form subintervals of waveforms (*windows*). That is, after a certain number of iterations the current window is split in two new windows, so that the first part will converge in a few more iterations and the second part will be less expensive than the original window. In [3] the windowing may force truncation of the last timestep in the window since integration steps are not allowed to pass a *window frame* except when a new window is started. We restart DAWRS in the next window with the non-truncated stepsize from previous window.

4 Convergence: Criteria and Techniques

If we define a map $W: z \mapsto z$, with $W(z(\tau))$ the solution of the decomposed system (3) in τ , then the WR iteration can be rewritten as a fixed point problem

$$z^j(\tau) = W(z^{j-1}(\tau)) \quad , \quad j = 1, 2, \dots \quad (6)$$

with $z^*(\tau) = W(z^*(\tau))$ the solution of the given system (1) in τ . The condition (2) controls the local error to generate an approximate sequence of W . Now, to check if this sequence is a converging sequence, the following criteria have to be satisfied in DAWRS. First we verify the timepoint convergence by the condition

$$\|y_i^j - y_i^{j-1}\| \leq \varepsilon_1 \quad , \quad i = 1, 2, \dots, p \quad (7)$$

for the current j th WR iteration. Whenever the condition (7) is satisfied for all timepoints in τ we check for the convergence of the waveform by condition

$$\frac{\rho_i}{1 - \rho_i} \|z_i^j - z_i^{j-1}\|_\infty \leq \varepsilon_2 < \varepsilon_1 \quad , \quad i = 1, 2, \dots, p \quad (8)$$

where $\|z(\tau)\|_\infty = \max_{t \in \tau} \|y(t)\|$, ε_1 and ε_2 scale the weight vector, w , to the allowed waveform tolerance, and ρ is an estimate of the convergence rate.

Allied to the splitting strategy we can drive the waveform sequence either by use of a *relaxation parameter*

²See [2] for its definition.

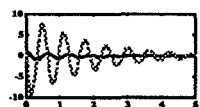
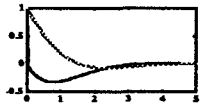
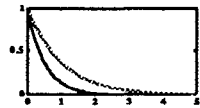
example	Case	Windows	WR Iterations	Timepoints	Residual Evaluations	Jacobian Evaluations	Rejected Timepnts	Information and Notes
<p>1</p> $\begin{aligned} \dot{y}_1 + 0.1y_1 - y_2 &= 0 \\ \dot{y}_2 + 100y_1 + y_2 &= 0 \\ y_1(0) = y_2(0) &= 1 \end{aligned}$ 	I	10	155	12251	24379	2708	720	<p>I: basic WR algorithm II: holding wave tail III: merging waveforms IV: prediction horizon V: timepoint insertion VI: fixed order 2 VII: fixed order 3 VIII: fixed order 4 IX: fixed order 5 X: maximum order</p>
	II	10	137	9186	18338	2050	630	
	III	10	137	9186	18338	2050	630	
	IV	10	94	5353	10685	954	207	
	V	10	94	5353	10685	954	207	
	VI	19	408	44881	89935	9666	4965	
	VII	10	115	10582	21171	1739	771	
	VIII	10	98	6123	12201	1144	329	
	IX	10	93	5850	11749	1155	338	
	X	10	95	6059	12102	1213	341	
<p>2</p> $\begin{aligned} \dot{y}_1 + y_1 + K \cos(t) X &= 0 \\ \dot{y}_2 + y_2 + K \sin(t) X &= 0 \\ X = y_1 \cos(t) + y_2 \sin(t) \\ y_1(0) = y_2(0) = 1, K = 999 \end{aligned}$ 	I	60	1252	80126	159777	14525	6677	<p>I-V use average BDF order. VI-X must be compared to IV (they use same conditions). Number of windows was fixed to ten. Example 2 and case VI in the example 1 use the splitting strategy.</p>
	II	53	1103	70490	140422	13618	6652	
	III	49	908	64105	127702	10861	5606	
	IV	24	441	15584	31019	3385	1427	
	V	22	419	15290	30349	3075	1348	
	VI	46	1126	42724	85084	9574	3289	
	VII	31	519	22511	44718	4894	1824	
	VIII	29	485	24905	49568	5145	2110	
	IX	26	461	17540	34780	3426	984	
	X	31	521	24394	48506	5108	2048	
<p>3</p> $\begin{aligned} \dot{y}_1 + 1002y_1 - 1000y_2 &= 0 \\ \dot{y}_2 - y_1 + (1 + y_2) y_2 &= 0 \\ y_1(0) = y_2(0) &= 1 \end{aligned}$ 	I	10	140	6860	13519	2294	624	<p>The global time interval for all simulations is [0, 1], and the accuracy is given by: Rel. Tolerance = 1×10^{-8} Abs. Tolerance = 0 Waveform Tol. = 1×10^{-7}</p>
	II	10	116	5843	11243	1753	489	
	III	10	116	5843	11243	1753	489	
	IV	10	53	1748	3089	692	57	
	V	10	53	1748	3089	692	57	
	VI	10	108	12097	23684	3371	1637	
	VII	10	71	3159	5875	999	178	
	VIII	10	68	2399	4374	888	96	
	IX	10	47	1600	2773	688	39	
	X	10	54	1956	3504	758	74	

Figure 2: DAWRS convergence characteristics. Modifications in the basic WR algorithm.

$\omega \in (0, 2)$, $z^j = (1 - \omega)z^{j-1} + \omega W(z^{j-1})$, where ω changes as the simulation proceeds, or introducing a variable local error criterion replacing the condition (2) by

$$M \|y_{n+1} - y_{n+1}^{(0)}\| \leq \xi \quad (9)$$

where $\xi \in [1, \infty)$ goes along with $\|z^j - z^{j-1}\|_\infty$ from a pre-specified value to one. While the relaxation parameter attempts to give more stability to the WR iteration when $\omega < 1$ and to accelerate the convergence when $\omega > 1$, the variable local error criterion prevents the integrator from using shorter timesteps.

A successful modification in the basic WR algorithm is that DAWRS retains useful information of past waveforms which makes the convergence faster than conventional approaches. First, DAWRS follows the neighbor integration history holding the *tail* of the neighbor waveform from the previous window and uses higher-order polynomials to interpolate the neighbor waveform at the beginning of the current window.

Second, instead of discarding the timepoints beyond the splitting point, we incorporate a *merging* strategy. When the first part of a split window converges DAWRS merges it into the second part, generating in this manner a more reliable initial guess to the next window. Because a window is only split after a reasonable number of iterations, the second part of a split window already has a good approximation of $W(z)$.

The third modification is the inclusion of a prediction horizon for the neighbor waveforms. When not available, DAWRS predicts the initial guess of the neighbor waveforms, $v_0(\tau)$, by polynomial extrapolation while the estimates for the leading error terms in the integrator form a decreasing sequence (see Section 2), and

from the point that does not match this condition to the end of the window $v_0(\tau)$ is kept constant.

Finally, DAWRS has a timepoint insertion into those neighbor waveforms that have less timepoints than necessary for the interpolation order. This insertion is done by polynomial interpolation using their own BDF data (before sharing the waveforms). This last modification is optional because the algebraic equations in the DAE's are not automatically satisfied at interpolated points. The Figure 2 shows three simple and representative examples of these modifications.

Since the waveforms have timesteps independent of each other, DAWRS has a neighbor interpolation formula to provide timepoints of the neighbor waveforms to the integration steps³. In Figure 2 we compare fixed order, maximum window order, and BDF⁴ average order interpolation polynomials. Although the example 3 shows a better performance when the order is fixed to 5, a performance degradation will likely occur in less active regions. Also, when the coefficients in the example 1, (0.1, -1, 100, 1), are replaced by (0.01, -100, 0, 100), a fixed order of 3 turns out to be better than 5. Thus, to monitor the activity changes and problem dependencies, a variable order given by the BDF average order seems to be a good choice.

5 Application Problems

We consider here two examples of distillation column networks, separating eight alcohols: methanol,

³In DAWRS the neighbor waveform, (5), does not include the derivatives, except for the first and last timepoints.

⁴The maximum BDF order was fixed to 5 in all simulations.

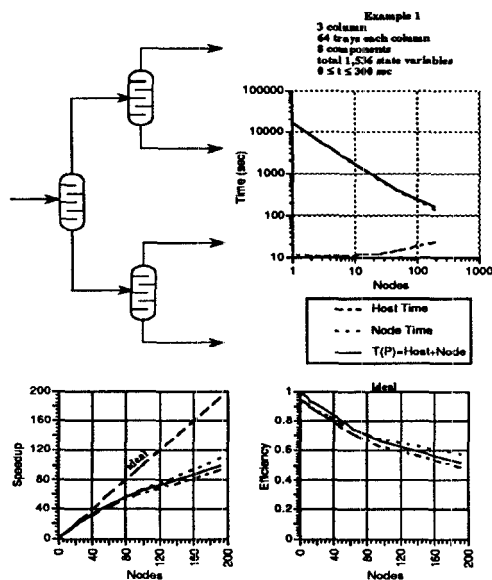


Figure 3: *Three-Column Network*. The real speedup, Sp , is represented by the solid line, and its lower and upper estimates are represented by the dashed lines.

ethanol, propan-1-ol, propan-2-ol, butan-1-ol, 2-methyl propan-1-ol, butan-2-ol, and 2-methyl propan-2-ol. The first network, Figure 3, has 1,536 state variables, and the second one, Figure 4, has 8,008. Each tray is initialized to a non-steady condition, and the system is relaxed to the steady state.

According to the distillation model utilized each tray has eight tightly coupled equations. The best grouping for this formulation was found to be one tray per subsystem. Subsystems bigger than that lose because of expensive calculation, and subsystems smaller than that lose by excessive WR iteration (around 7-10 against 4-5) and massive waveform exchange. Also, we can distinguish four types of “trays” with significantly different activities (reboiler, condenser, feed tray, and other trays). Therefore, we can expect a load balancing problem. The max/min node CPU-time ratio was found to be between 1.5-3 when the work load is even in terms of number of equations per node. We have reduced this ratio to 1.1-1.3 by evening the work load for the residual and Jacobian evaluation, and also taking into account the size of each subsystem. The Figures 3 and 4 show the obtained speedup on a 192-node Symult s2010 multicomputer⁵. The tolerances indicated in Figure 2 were used in these simulations. The dashed lines in the Speedup graphics are lower and upper estimates for the relative speedups given by

$$Sp^U(P) = \frac{CPU - Host(P)}{Node(P)}, \quad Sp^L(P) = \frac{CPU}{Node(P) + Host(P)}$$

where $CPU = Host(P) + \sum_{i=1}^P Node_i(P)$, $Host(P)$ is the

⁵Qualitatively the same results were obtained on a 64-node Intel iPSC860 multicomputer, with all time reduced by a factor 4.5-5. Thus the behavior seems independent of architecture.

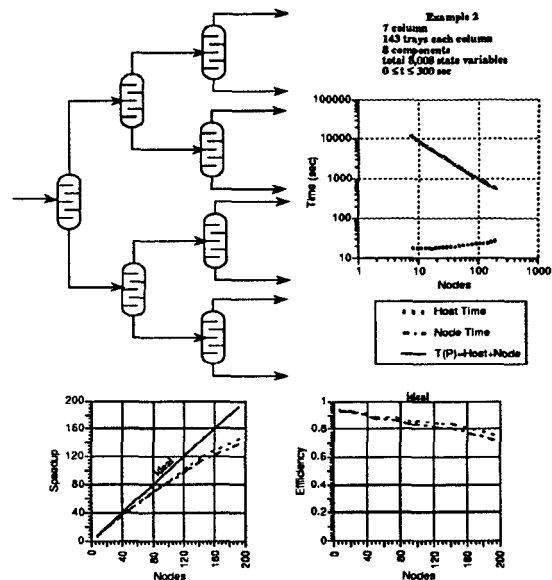


Figure 4: *Seven-Column Network*. The dashed lines represent the lower and upper estimates for the relative speedup, $Sp(P)$, and efficiency, $\eta(P)$.

host time, $Node(P)$ is the maximum node CPU-time, and P is the number of processors. Due to memory limitation, the real relative speedup and efficiency,

$$Sp(P) = \frac{T(1)}{T(P)}, \quad \eta(P) = \frac{Sp(P)}{P} \quad (10)$$

where $T(P) = Node(P) + Host(P)$, could only be calculated for the three-distillation-column network, represented by the solid line in Figure 3.

6 Conclusion and Future Work

We have showed significant gains in performance as a result of our new approaches to manipulation of the waveforms. In future work we intend to provide new partitioning strategies, such as new ordering schemes, and a more detailed grouping analysis, in order to achieve higher convergence rates. Experimental results with DAWRS, applied to distillation columns, already show that the WR method is a strong candidate as a concurrent flowsheeting simulation methodology. We expect to apply the technique to many other interesting applications in chemical engineering, and to make time comparisons with sequential algorithms and other concurrent methods (e.g., direct methods).

References

- [1] A. Skjellum, *Ph.D. Thesis*, Caltech, 1990.
- [2] K.E. Brenan, S.L. Campbell, L.R. Petzold, *Numerical Solution of I.V.P. in DAE's*, North-Holland, 1989.
- [3] S. Mattisson, *CONCISE: A Concurrent Circuit Simulation Program - Ph.D Thesis*, Lund University, Sweden, 1986.
- [4] A. Skjellum, M. Morari, S. Mattisson, *Proc. of the 3rd Conf. on Hypercube Concurrent Computers and Appl.*, Vol.2, pp.1062-71, ACM Press, 1988.
- [5] E. Lelarsmoe, *Ph.D. Thesis*, UC Berkeley, 1982.
- [6] L. Peterson, *A Study of Convergence-Enhancing Techniques for Concurrent WR*, Lund University, Sweden, 1989.
- [7] R. Tarjan, *SIAM J. Comput.*, Vol.1, No.2, pp.146-160, 1972.