FEMSYS – An explicit finite element code for non-linear dynamic structural analysis: a time integration strategy

Carlos Alberto Nunes Dias, Larissa Driemeier
Group of Solid Mechanics and Structural Impact,
University of São Paulo, São Paulo/SP – Brazil
Marco Lucio Bittencourt
Department of Machine Design, University of Campinas, Campinas/SP – Brazil
Marcílio Alves
Group of Solid Mechanics and Structural Impact,
University of São Paulo, São Paulo/SP – Brazil

Abstract

This article describes the general strategy adopted in an explicit finite element code for time integration. It is introduced some details of the code and implementation strategies to handle non-linear response of 3D structures. Next, an integration algorithm within the framework of control theory is outlined, with some examples indicating the merits of the method. A discussion is presented in the context of the development of an open Brazilian finite element programme for use in the development of material models and finite elements.

1 Introduction

The analysis of structures subjected to loads of high intensity and short duration is one of the most challenges areas in Solid Mechanics. Phenomena like material visco-plastic behaviour, inertia effects, contact, large rotations and deformation and inertia effects add to other complex issues such as complex boundary conditions and initial conditions difficult to be known. No wonder that analytical methods cannot take all this effect into account, so a detailed analysis can only be performed using numerical methods.

There are a few numerical techniques capable of handling, albeit in different ways, the aspects listed above. The simulation in Figure 1, for instance, of a plate impacted by a soft cylinder, was performed using the so called SPH method, Smooth Particle Hydrodynamics.

But no doubt that, chiefly among the numerical techniques, it is the Finite Element Method, FEM, the most mature technique for structural analysis. This technique acquired a technological impulse with the continuous demand of the industry for user-friendly software. Nowadays, a dozen of packages

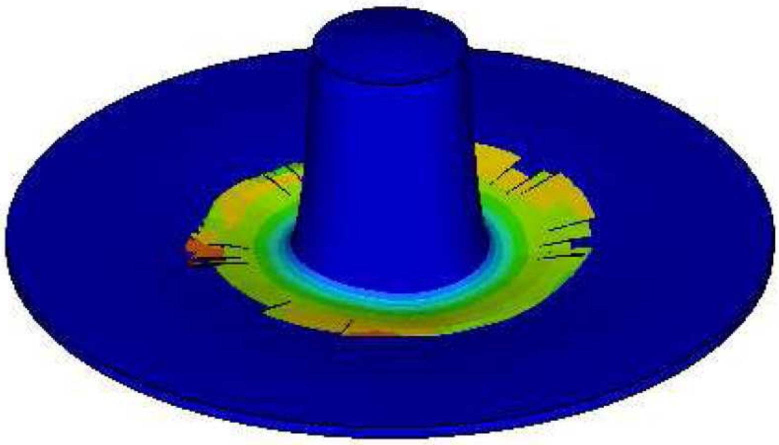


Figure 1: A composite plate impacted by a soft cylinder analysed using the SPH method.

are available capable of handling the issues raised above.

The finite element market is worldwide dominated by powerful software such as ABAQUS, ANSYS, DYTRAN, NASTRAN , LS-DYNA, RADIUS. It is interesting to note that Brazil is more and more a product developer, with important design activities in the automotive and aircraft industries, naval architecture and related products in general, to name a few. Many of these branches use intensively the FEM. It seems that, in Brazil, there is no systematic effort to bring out, if not to the market but at least to the academic community, a finite element package capable of handling large scale non-linear dynamic problems. Such is the objective of the authors, as far as they are developing a code which will allow other researchers to incorporate new finite elements and constitutive laws.

In this article, we outline an Explicit FEM code starting by giving an overall idea of its basic structure. We then show that the integration algorithm should be robust and efficient, so section 3 describes an explicit algorithm, with section 4 presenting some basic examples. Section 5 discusses some aspects of the programme, of pre and post model handling, of high order finite elements, of contact and of failure criteria, with section 6 closing the article.

2 Basic code

The FEMSYS code is built around functions. At the moment, there are two versions, one written in Matlab code and the other in Fortran. Matlab was chosen for developing the routines and testing them, while Fortran, being more robust, handles larger problems more efficiently. However, a new Matlab Tool Box allows parallel processing, which greatly simplifies parallel computing. Hence, it is likely that the final version 0 of the code will be available as a Matlab compiled version. The fact that a Matlab platform is being used reduces further developing time and programming resources, allows more expeditiously documentation and facilitates the use of the programme as a teaching tool.

In cases of dynamic beam response or general structural impact, the structure response is obtained by time integration of the equilibrium equations. In this case, it is meaningful to talk about implicit and explicit time integration.

For quasi-static problems, the finite element problem seeks to find the structure configuration, u, via

$$u(x) = K^{-1}(x)F(x)$$

We see that the approach requires to find the inverse of the stiffness matrix, K, for each displacement of force, F, increment. This is in frank contrast with dynamic problems, whose general equation to be solved is

$$M(x)\ddot{x} + C(x,t)\dot{x} + K(x,t)x = F(x,t)$$

Here, M and C are the mass and damping matrices. Observe that it is possible to obtain the displacement, velocity and acceleration of the structure by marching forward in time with no inversion of the stiffness matrix. Hence, explicit methods can be of advantage in dynamic problems whose overall time duration is much less than the first natural period of the structure under analysis. Also, to allow uncoupling of the system above, the mass matrix should be diagonal. It is clear from above that, crucial to the efficiency of the process is the time integration algorithm. This lead the authors to develop an integration algorithm meant to be efficient, robust and with little user interference.

3 An explicit integration algorithm

Explicit algorithms are unstable in the sense that the time integration of the equilibrium equation governing the behaviour of a structure may not converge to a physical solution as time progresses. In fact, there is no guarantee that even implicit methods are stable in the context of nonlinear structural analysis [1]. It is a common practice to define the minimum time step in explicit algorithms based on the highest natural frequency of the finite element model, leading to time steps of the order of micro-seconds.

Explicit integration has the chief advantage of solving an uncoupled algebraic system. This leads to smaller processing time at each iteration. Also, it is not necessary to work with the stiffness matrix directly; instead, one can use the internal forces.

Clearly, it is interesting to have different time steps in the time integration of the equilibrium equations. Hence, if no stability is detected in the procedure, one could increase the time step so decreasing the overall processing time when bearing in mind that a normal dynamic non-linear transient analysis comprises thousands of iterations.

In our FE programme, we adapted a Proportional, Integral, Derivative, PID, control [2] as a basic tool to change the time step as the simulation progresses, whose analogy with the level of a water tank, as in Figure 3, amounts to its understanding. The main tank is fed by a continuous flux of inputs and feeds the secondary tank. The aim is to fill in the secondary tank in the least possible time, which requires a high main tank level. However, due to the road and driver manoeuvres, the fluid can spill out of the main tank, which can be avoided by setting a maximum reference level. But this level should change continuously to take advantage of the stability of the driving solution process, should it occur. With the PID control, it is possible to fine tune the time step by defining only three equations which, based on the difference between integration error and tolerance, define the actions of the PID control, so establishing a more suitable time step.

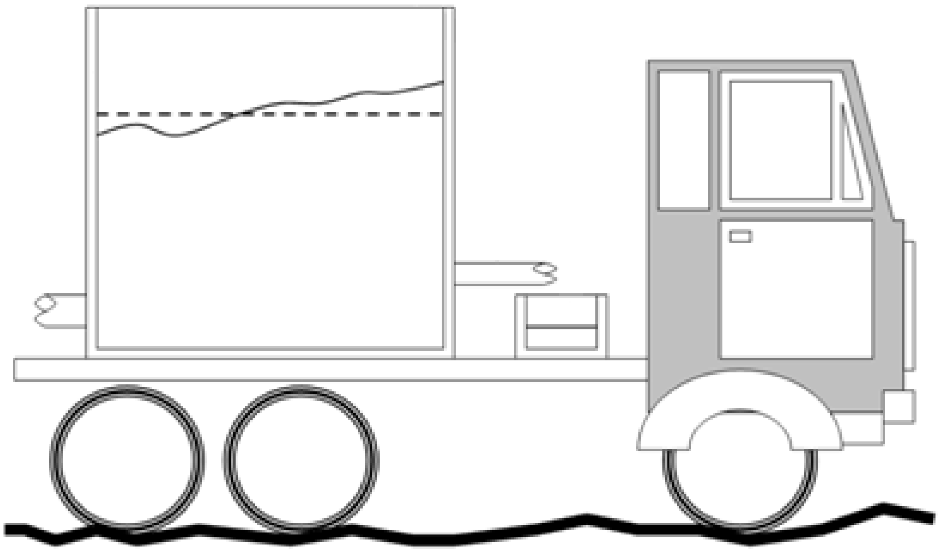


Figure 2: Understanding the PID control used in FEMSYS.

3.1 Linear acceleration (supply) method

A major difference between the central difference method, not detailed here, and the Linear Acceleration Method, LAM [3] is that, in the later, equilibrium is imposed at the end of the interval. In this case, the kinematic parameters of the system are yet to be known, so it is necessary to adjust the equilibrium via an interactive process. This increases a little the processing time but improves dramatically the quality of the results.

In order to describe the LAM, let us consider a single degree of freedom model, with the acceleration varying linearly in the interval $[t;\, t + dt]$. For $0 \leqslant \tau \leqslant dt$ one can write

$$\ddot{x}(\tau) = A\tau + \ddot{x}_n \tag{1}$$

$$\dot{x}(\tau) = A\tau^2/2 + \ddot{x}_n\tau + \dot{x}_n \tag{2}$$

$$\bar{x}(\tau) = A\tau^3/6 + \ddot{x}_n\tau^2/2 + \dot{x}_n\tau + x_n \tag{3}$$

with x_n, $\dot{x}_n$and $\ddot{x}_n$ being the displacement, velocity and acceleration, respectively, already known in the beginning of the interval. A is a constant, which can be determined by enforcing equilibrium at $t + dt \Leftrightarrow \tau = dt$, and represents the gradient of the acceleration. Equilibrium and kinematic relations lead to

$$x^i_{n+1} = x_n + dt\dot{x}_n + (dt^2/2)\ddot{x}_n + (dt^3/6)A^{i-1}_n \tag{4}$$

$$\dot{x}^i_{n+1} = \dot{x}_n + dt\ddot{x}_n + (dt^2/2)A^{i-1}_n \tag{5}$$

$$\ddot{x}^i_{n+1} = m^{-1}(F^{ext}_{n+1} - c\dot{x}^i_{n+1} - P^i_{n+1}) \tag{6}$$

$$A^i_n = (\ddot{x}^i_{n+1} - \ddot{x}_n)/dt \tag{7}$$

with i standing for current iteration, F^{ext}_{n+1} external loads and P^i_{n+1} internal elastic forces in position x^i_{n+1} , both at $t = \tau$. The iterative process starts with $A^{i-1}_n = A^I_{n-1}$ and finishes when the acceleration rate, $\left\|A^i_n - A^{i-1}_n\right\| / \left\|A^i_n\right\| \leqslant tolA$, attains a value lower than a pre-defined tolerance, *tolA*. Once this tolerance is attained, after I iterations, the kinematic parameters are given by

$$x_{n+1} = A^I_n dt^3/6 + \ddot{x}_n dt^2/2 + \dot{x}_n dt + x_n \tag{8}$$

$$\dot{x}_{n+1} = A^I_n dt^2/2 + \ddot{x}_n dt + \dot{x}_n \tag{9}$$

$$\ddot{x}_{n+1} = A^I_n dt + \ddot{x}_n \tag{10}$$

This method equally applies to a system of several degrees of freedom, with the remark that the mass matrix is diagonal so that equation (6) is uncoupled, which justifies the adjective *explicit* to the method.

3.2 PID control

For linear problems, the critical time step in explicit time integration procedures can be written as

$$dt \leqslant dt_{cr} = \frac{2}{\omega_{n_{\max}}} \tag{11}$$

with $\omega_{n_{max}}$being the maximum natural frequency of the discrete model. Equation (11) implies that, contrary to the common sense, finer meshes leads to smaller time integration teps. A remark is that the maximum natural frequency, although an eigen value of the mathematical problem, has no physical meaning.

Noels et al. [4] present a complicate strategy for the establishment of the time step. Here, this task is greatly simplified by using the adaptative control, which can even be used for low velocity problems, i.e. quasi-static problems.

The time step, dta, in our strategy is calculated according to

$$dta = (2^{a_T})dtb \tag{12}$$

with a_T a control parameter which increases ($a_T > 0$) , decreases ($a_T < 0$) or maintains ($a_T = 0$) the time step, i.e. a_T scales the time step.

The initial time step is set as the critical time step, the later based on the maximum natural frequency of the initial structure configuration.

The control action a_T can be split by the sum of the proportional a_P, integral a_I and derivative a_D actions [2], i.e.

$$a_T = a_P + a_I + a_D \tag{13}$$

with

$$a_P^{n+1} = g_P(V_n - R_n) \tag{14}$$

$$a_I^{n+1} = a_I^n + f_{aI}(V_n - R_n) \tag{15}$$

$$f_{aI} = g_P T_S / T_I \tag{16}$$

$$a_D^{n+1} = f_{aD} a_D^n - f_{cD}(V_n - V_{n-1}) \tag{17}$$

$$f_{aD} = T_D/(T_D + NT_S) \tag{18}$$

$$f_{cD} = g_P N f_{aD} \tag{19}$$

Here, $V_n$is the variable to be controlled and kept close to the reference R_n, with a gain g_P. T is the time, with indexes I, D and S standing for integral, derivative and sampling, N ($3 \leqslant N \leqslant 20$), limits the high frequency filter. It follows that the new time step is given by

$$dt^{n+1} = 2(a_P^{n+1} + a_I^{n+1} + a_D^{n+1})dtb \tag{20}$$

These three gains are adjusted in the present approach just once. This strategy removes from the analyst the burden of choosing, or guessing, the best values. The disadvantage is that, for each time step, the optimum control parameters should vary.

Of course there is an integration error, which we evaluate via

$$e(t) = dt^2(\Delta\ddot{Q})_{\max}/Q_{est} \tag{21}$$

with dt being the time step value, $(\Delta\ddot{Q})_{\max}$ the maximum acceleration variation among all the degrees of freedom of the structure and Q_{est} the maximum structure displacement under a quasi-static load. This measure of error varies with time but it was found that its change is more dramatic when the time step changes, hence it is an interesting measure to work with.

To obtain the amplitude and frequency of $e(t)$ we work with its last four values, e_n , e_{n-1}, e_{n-2} and e_{n-3} at time t_n, t_{n-1}, t_{n-2} and t_{n-3}. The error frequency comes from

$$\omega^2(t) = \left| D \left\langle \frac{\ddot{e}(t)}{e(t)}; \frac{\dddot{e}(t)}{\dot{e}(t)} \right\rangle \right| \tag{22}$$

with $D\langle \bullet ; \bullet \rangle$ representing the minimum value between $\ddot{e}/e$ and $\dddot{e}/\dot{e}$. The amplitude of the error is given by

$$A_n = \sqrt{e_n^2 + \dot{e}_n^2 / w_n^2} \tag{23}$$

which is filtered according to

$$\bar{A}_n = \left(\sum_{i=n-nf+1}^{n} A_i \right) / nf \tag{24}$$

$$E_I^n = \bar{\bar{A}}_n = (\bar{A}_n + p_A \bar{\bar{A}}_{n-1})/(1 + p_A) \tag{25}$$

being E_I^n the final integration error to be used in the time step control algorithm. Here, p_A is a factor which aims to limit sudden changes in the error amplitude.

For the LAM, the current error is penalised, i being the number of iterations, by

$$\bar{e}_n = [1 + (i/10)^4] e_n \tag{26}$$

so that, for i<<10, $\bar{e}_n \cong e_n$. For $i \approx 10$ the error is penalised by 100% and increases for $i > 10$. Hence, when the number of iterations grows, the time step decreases, and vice versa.

It is useful to note that the integration error exhibits an oscillation related to instability. Accordingly, when the integration is stable, the integration error oscillates with a period much longer than the time step, i.e. closer to the first natural periods of the structure. Conversely, close to instability, the period of error oscillation decreases substantially, getting closer to time integration step. Hence, it is possible to obtain some hints about the stability of the solution by observing the integration error behaviour. This, in turn, assists one to alter the integration time step, which in the present programme is performed automatically.

There are other issues to be considered, like the initial time step to be used and the strategy to be adopted for the change in the error tolerance as instability approaches, but they are not considered here.

4 Examples

The integration method LAM is explored in this section together with the automatic time step control based on PID. We start by analyzing a linear single degree of freedom system excited by a harmonic function. Tables 1 and 2 show the results when using the Central Difference, CDM, and LAM methods, with Table 3 summarizing the CPU time.

In order to explore different numerical integration difficulties, we adopt three damping factors of 1, 10 e 50 %, and three dynamic configurations of high velocity (impact), medium velocity (resonance) and low velocity (quasi-static equilibrium) via the adoption of the three excitation to natural frequency ratios of 10, 1 and 0.1, respectively.

Table 1: Response of one degree of freedom system calculated according to the Central Difference methods. $m = 1$, $k = 1$, integration time $4\pi/\omega$, load amplitude 1, initial time step 10ms, sampling time $T_s = 10$ms, zero initial conditions, proportional gain $1/R_i$, integral time $100T_s$, derivative time $5T_s$.

ω/ω_n	ζ	Time step				Integration error				Error (%) Analyt.
		Initial E-2(s)	% of initial time step			Initial toler. E-6 (s)	% of initial toler.			
			min	med	max		min	med	max	
10	0.01	1.5708	40.25	51.69	99.02	3.5076	92.50	151.83	206.24	0.71
	0.1	=	41.64	51.89	99.01	3.4948	92.47	151.47	202.12	0.68
	0.5	=	36.19	51.18	99.00	3.4450	92.39	153.13	207.14	0.53
1	0.01	2.0000	48.78	71.75	100.00	6.3400	80.94	108.86	133.93	5.50
	0.1	=	53.19	77.30	100.00	6.1950	75.37	107.16	143.14	4.32
	0.5	=	81.09	96.80	112.77	6.1182	75.10	100.92	119.15	2.00
0.1	0.01	2.0000	66.06	111.84	137.84	0.6372	72.41	99.33	145.87	8.63
	0.1	=	91.76	265.17	532.17	0.6273	68.35	90.97	134.09	1.13
	0.5	=	93.08	434.10	769.73	0.6410	21.08	81.19	164.5	1.28

From the tables, the Central Difference method is less accurate and not recommended for medium and low velocities. On the other hand, the Supply method is adequate for all the three situations explored in these examples. This is so because the Supply method is capable of adjusting itself to these rather different situations, although to the expense of CPU time, as seen in Table 3. Figures 4, 5 and 6 present the time variation of various solution parameters for the cases of high, medium and low velocity loading, respectively.

Next, we apply both integration methods to a non-linear single degree of freedom problem. The system displaces backwards with greater amplitude than when moving forwards with high velocity. This causes sudden changes in the system behavior, which should be handled by the integration algorithm.

Figure 7 and 8 depict the behavior of the system with time, as calculated using the central difference and the linear acceleration methods, respectively. The various parameters of the system used in the computation are given in the caption.

Table 2: As in Table 1 but for the LAM method.

ω/ω_n	ζ	Time step				Integration error				Error (%) Analyt.
		Initial E-2(s)	% of initial time step			Initial toler. E-6 (s)	% of initial toler.			
			min	med	max		min	med	max	
10	0.01	1.5708	42.13	53,34	99.50	3.8265	95.47	149.54	191.83	0.090
	0.1	=	32.86	50.98	99.52	3.8760	95.19	154.81	216.92	0.100
	0.5	=	1.391	19.81	99.51	0.5279	60.72	156.31	610.45	0.144
1	0.01	2.000	43.37	74.83	100	6.8970	79.01	108.34	150.84	0.007
	0.1	=	60.32	81.00	100	7.0771	79.03	106.06	126.34	0.006
	0.5	=	100.00	120.24	132.23	13.452	78.31	94.48	105.35	0.005
0.1	0.01	2.000	73.26	145.32	222.17	0.6871	72.59	97.40	136.80	0.014
	0.1	=	93.02	323.27	732.31	0.7204	52.10	86.39	115.62	0.0009
	0.5	=	98.49	474.96	705.39	0.9750	21.50	79.86	130.84	0.0002

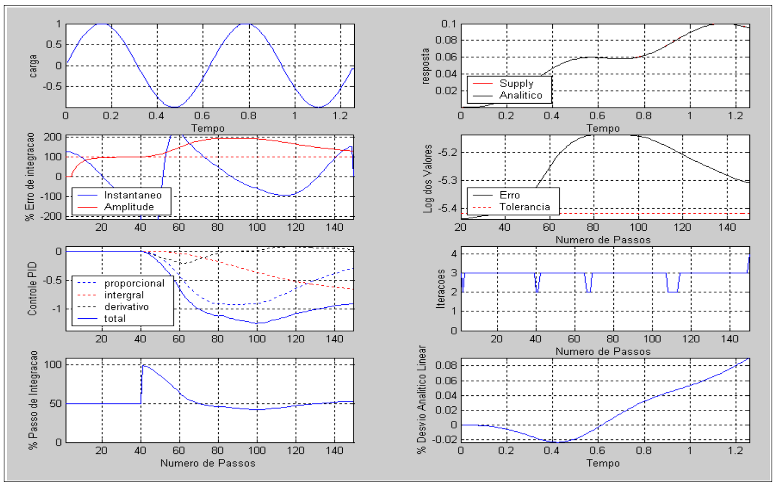


Figure 3: Various parameters of the response of the one degree of freedom system: high velocity (impact).

Table 3: Comparison of processing time (s) for Central Difference, CDM, and Linear Acceleration, LAM, methods. Pentium IV 2533MHz, MatLab, Windows 2000.

ω/ω_n	ζ	LAM	CDM
10	0.01	0.438	0.172
	0,1	0.218	0.156
	0.5	0.515	0.172
1	0.01	0.984	0.687
	0,1	0.937	0.641
	0.5	0.672	0.531
0.1	0.01	7.453	7.484
	0,1	2.578	2.031
	0.5	1.812	1.157
TOTAL		15.61	13.03

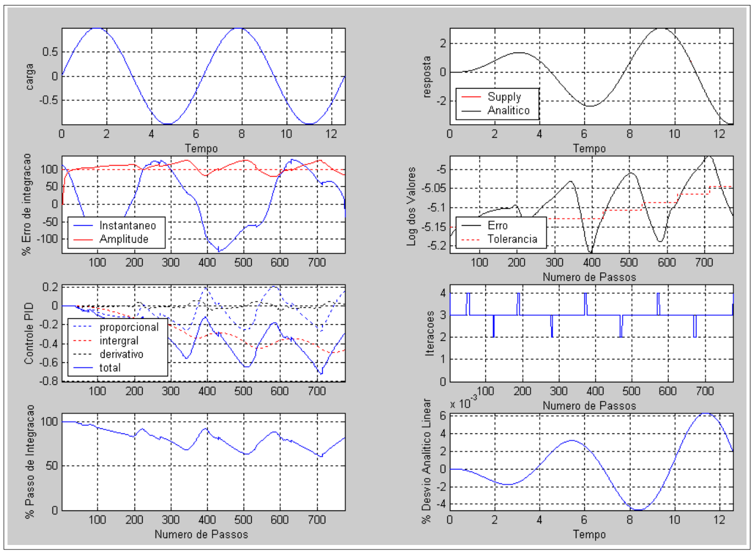


Figure 4: Various parameters of the response of the one degree of freedom system: medium velocity (resonance).

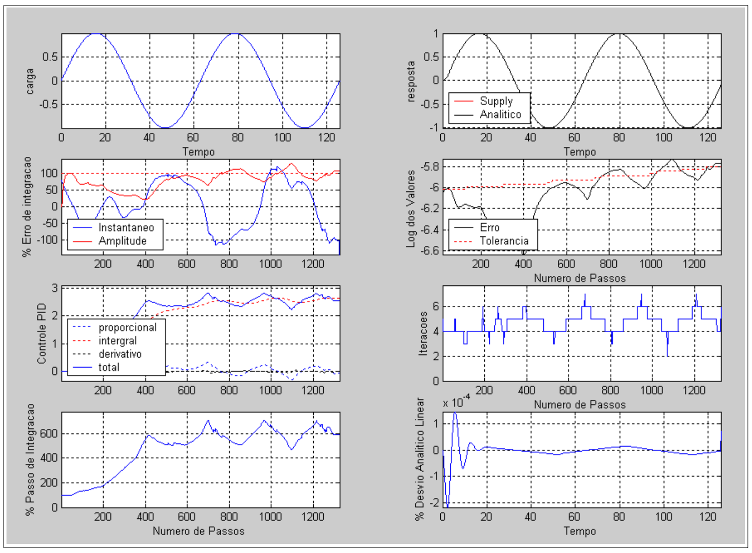
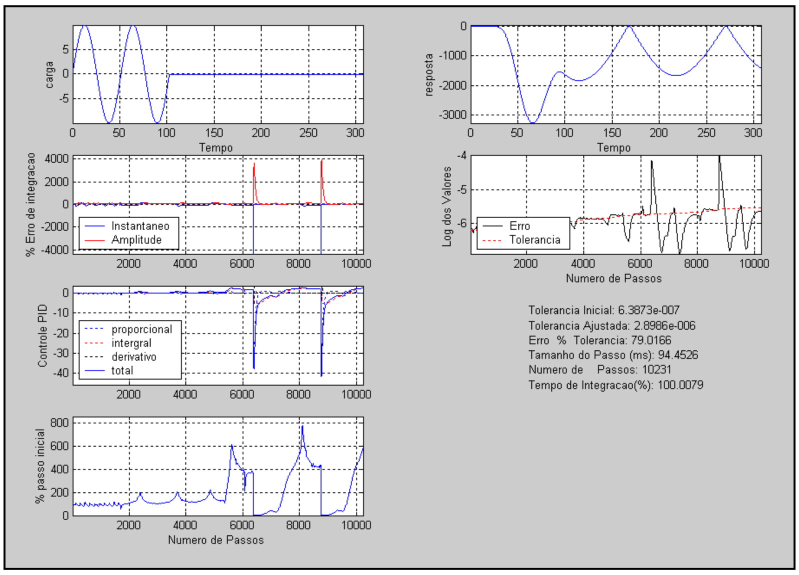
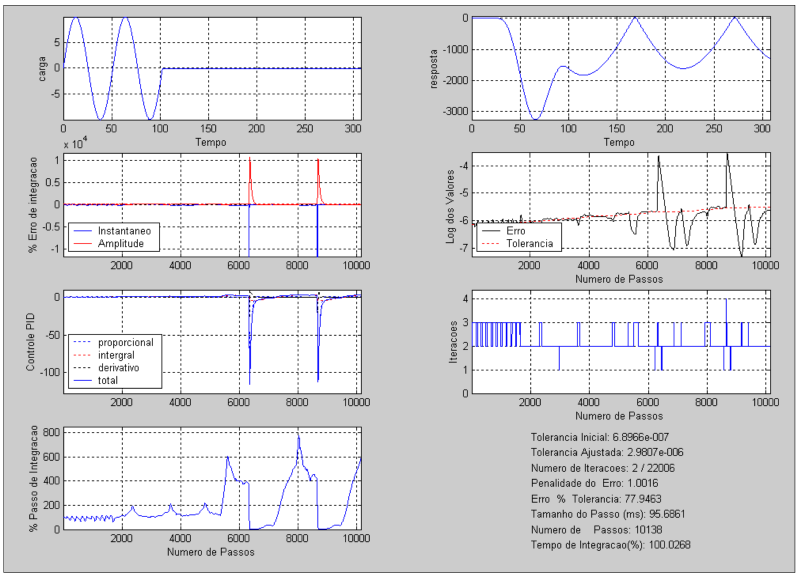


Figure 5: Various parameters of the response of the one degree of freedom system: low velocity (quasi-static).



```
Tolerancia Inicial: 6.3873e-007
Tolerancia Ajustada: 2.8986e-006
Erro  %  Tolerancia: 79.0166
Tamanho do Passo (ms): 94.4526
Numero de    Passos: 10231
Tempo de Integracao(%): 100.0079
```

Figure 6: Non-linear system response according to the central difference method. m=1, $k_{forward} = 1.5$, $k_{backward} = 0.001$, $\xi = 0.1\%$, load amplitude = 10, load frequency = 0.1225, load duration = 102.60, initial conditions zero.



```
Tolerancia Inicial: 6.8966e-007
Tolerancia Ajustada: 2.9807e-006
Numero de Iteracoes: 2 / 22006
Penalidade do  Erro: 1.0016
Erro  %  Tolerancia: 77.9463
Tamanho do Passo (ms): 95.6861
Numero de    Passos: 10138
Tempo de Integracao(%): 100.0268
```

Figure 7: Non-linear system response according to the linear acceleration method. System parameters as in Figure 7 plus dtb = 1.633e-2, Ri = 6.8966e-7; Ts = dtb, proportional gain = 1/Ri, integral time = 100Ts, derivative time = 5Ts.

5 Closure

An explicit finite element code is introduced in the hope that can serve for other researchers to use for implementing new material models and finite elements. This will lead to further development of the code, equipping the academia with a useful tool for analysis, research and teaching.

The explicit code is underpinned by time integration of the equilibrium equations. This point to an efficient algorithm, here described. When comparing with the classical Central Difference Method, the Linear Acceleration Method outlined here is a more robust tool to deal with the time integration. As a novelty, we offer an automatic time step control, which sets longer or shorter time steps as the solution progresses.

There are other aspects of an explicit code, not addressed here, which requires the user input, such as hour-glass parameters. But, at least for the integration algorithm, the method frees the user of input figures whose precise effect is not well known. Such a strategy has been tested for more complex models, comprising hundreds of finite elements, and the results are all promising.

References

[1] Bath, K.J., *Finite Element Procedures.* Prentice Hall: New Jersey, 1996.

[2] Caon, J.R., *Controladores PID Industriais com Sintonia Automática por Realimentação a Relê*. Master's thesis, USP/São Carlos, 1999.

[3] Francisco, C.A. & Nunes Dias, C.A., Introdução ao estudo de vibrações sub-harmônicas em embarcações amarradas. *Revista Brasileira de Engenharia Caderno de Engenharia Naval*, **5(1)**, 1988. ISSN 0102-2679.

[4] Noels, L., Stainer, L. & Ponthot, J.P., Self-adapting time integration management in crash-worthiness and sheet metal forming computation. *Int J of Vehicle Design*, **30(1/2)**, 2002.