Available online at www.sciencedirect.com

ScienceDirect

Fusion Engineering and Design 83 (2008) 188–192

Fusion Engineering and Design

www.elsevier.com/locate/fusengdes

Adaptation of the MAST passive current simulation model for real-time plasma control

G.J. McArdle [*], D. Taylor

EURATOM/UKAEA Fusion Association, Culham Science Centre, Abingdon, Oxfordshire OX14 3DB, UK

Received 1 June 2007; received in revised form 22 October 2007; accepted 1 November 2007
Available online 4 March 2008

Abstract

Successful equilibrium reconstruction on MAST depends on a reliable estimate of the passive current induced in the thick vacuum vessel (which also acts as the load assembly) and other toroidally continuous internal support structures. For the EFIT reconstruction code, a pre-processing program takes the measured plasma and PF coil current evolution and uses a sectional model of the passive structure to solve the ODEs for electromagnetic induction. The results are written to a file, which is treated by EFIT as a set of virtual measurements of the passive current in each section.

However, when a real-time version of EFIT was recently installed in the MAST plasma control system, a similar function was required for real-time estimation of the instantaneous passive current. This required several adaptation steps for the induction model to reduce the computational overhead to the absolute minimum, whilst preserving accuracy of the result. These include:

- conversion of the ODE to use an auxiliary variable, avoiding the need to calculate the time derivative of current;
- minimise the order of the system via model reduction techniques with a state-space representation of the problem;
- transformation to eigenmode form, to diagonalise the main matrix for faster computation;
- discretisation of the ODE;
- hand-optimisation to use vector instruction extensions in the real-time processor;
- splitting the task into two parts: the time-critical feedback part, and the next cycle pre-calculation part.

After these optimisations, the algorithm was successfully implemented at a cost of just 65 μs per 500 μs control cycle, with only 27 μs added to the control latency. The results show good agreement with the original off-line version. Some of these optimisations have also been used subsequently to improve the performance of the off-line version.

© 2007 The European Commission. Published by Elsevier B.V. All rights reserved.

Keywords: Induced current; Model reduction; Discretisation; State space; Plasma control

1. Introduction

The main tool for plasma equilibrium reconstruction on MAST is EFIT [1], a free-boundary Grad Shafranov equilibrium solver. This code fits the plasma current distribution to a constrained current profile function that is consistent with the measured signals from the flux and field diagnostics, the measured coil currents and the Grad Shafranov equation.

The "coil set" as configured in EFIT (see Fig. 1) actually consists of both active poloidal field coils and passive (toroidally continuous) vessel elements and support structures, but only the active currents are measured. The MAST vacuum vessel also serves as the load assembly, so it is much thicker than is needed for a mere vacuum boundary and the total-induced passive current can be of the order of 100 kA. Since this is significant enough to adversely affect the reconstruction of the plasma equilibrium, the EFIT code needs to be given an estimate of the induced passive currents and their distribution so that it can take them into account.

1.1. Passive current simulation model

In many tokamaks the passive current is simply fitted resistively to the voltage measured from several loops attached to

[*] Corresponding author. Tel.: +44 1235 466729; fax: +44 1235 466379.
E-mail address: graham.mcardle@ukaea.org.uk (G.J. McArdle).

0920-3796/$ – see front matter © 2007 The European Commission. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.fusengdes.2007.11.003

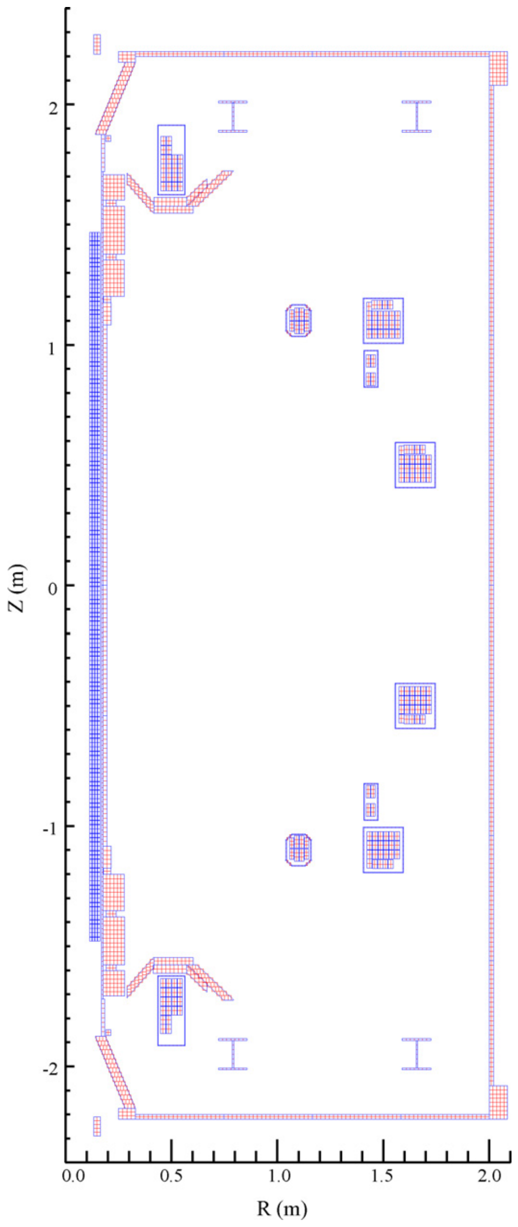

Fig. 1. Poloidal cross-section of the MAST conductor model used by EFIT.

the passive structures. This is not currently possible for MAST because such measuring loops are only available on the centre tube of the vessel. The induced passive current is therefore estimated by solving the induction equation for the passive circuit model as a ‘pre-processing’ stage before EFIT is run.

If $\mathbf{I}_\mathrm{v}$ is a vector of vessel (and other passive) element currents, $\mathbf{I}_\mathrm{c}$ is the vector of active coil currents (here we include plasma current modelled as a diffuse ‘coil’ with a generic current distribution), $\mathbf{R}_\mathrm{v}$ is a diagonal matrix of vessel element resistances, $\mathbf{L}_\mathrm{v}$ the inductance matrix of the vessel elements, $\mathbf{M}_\mathrm{vc}$ is the mutual inductance matrix between the vessel elements and the active coils, then since there is no external voltage applied to the passive conductors, the induction equation is

$$(V =)\, 0 = \mathbf{R}_\mathrm{v} \cdot \mathbf{I}_\mathrm{v} + \mathbf{L}_\mathrm{v} \cdot \dot{\mathbf{I}}_\mathrm{v} + \mathbf{M}_\mathrm{vc} \cdot \dot{\mathbf{I}}_\mathrm{c} \tag{1}$$

which can be rearranged to the following ordinary differential matrix equation:

$$\dot{\mathbf{I}}_\mathrm{v} = (-\mathbf{L}_\mathrm{v}^{-1} \cdot \mathbf{R}_\mathrm{v}) \cdot \mathbf{I}_\mathrm{v} + (-\mathbf{L}_\mathrm{v}^{-1} \cdot \mathbf{M}_\mathrm{vc}) \cdot \dot{\mathbf{I}}_\mathrm{c} \tag{2}$$

The passive current estimator for EFIT solves this ODE using the recorded data for $\mathbf{I}_\mathrm{c}$.

1.2. The need for a real-time version

A real-time version of the EFIT code, rtefit [2], was developed by General Atomics for use in their Plasma Control System (PCS, [3]). The same software infrastructure is also in use on the MAST PCS [4], and rtefit has recently been installed and configured in MAST PCS.

The runtime of the EFIT induction model is longer than the pulse duration (although this includes loading of files and saving of results). An equivalent version of this passive current simulation model was required for PCS to be used with rtefit. For this version to be useful in real time it was required to provide an instantaneous estimate of the passive currents for each control cycle, ready to be used by rtefit for real-time boundary reconstruction.

2. Optimisations for real-time operation

The algorithm for passive current estimation was adapted for real-time application using several optimisations as presented below.

2.1. Change of variables to avoid differentiation

The source term in Eq. (2) is the time derivative of active currents. However, the available measurement is normally $\mathbf{I}_\mathrm{c}$, and it is wasteful to differentiate the measured value only to integrate it again when solving the ODE. Introducing the variable $\mathbf{x}$, where

$$\mathbf{I}_\mathrm{v} = \mathbf{x} - \mathbf{L}_\mathrm{v}^{-1} \cdot \mathbf{M}_\mathrm{vc} \cdot \mathbf{I}_\mathrm{c} \tag{3}$$

and substituting this into Eq. (2) we get:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = (-\mathbf{L}_\mathrm{v}^{-1} \cdot \mathbf{R}_\mathrm{v}) \cdot \mathbf{x} + (\boldsymbol{L}_\mathrm{v}^{-1} \cdot \boldsymbol{R}_\mathrm{v} \cdot \boldsymbol{L}_\mathrm{v}^{-1} \cdot \boldsymbol{M}_\mathrm{vc}) \cdot \mathbf{I}_\mathrm{c} \tag{4}$$

i.e. it is no longer necessary to differentiate the measured coil current $\mathbf{I}_\mathrm{c}$, not even to recover $\mathbf{I}_\mathrm{v}$ from $\mathbf{x}$ with Eq. (3). Eqs. (3) and (4) can be expressed in standard state-space notation:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \tag{5}$$

$$\mathbf{y} = \mathbf{C} \cdot \mathbf{x} + \mathbf{D} \cdot \mathbf{u} \tag{6}$$

with obvious variable substitutions.

2.2. Model reduction

The vector $\mathbf{x}$ is a state vector of internal dynamical state variables for the system with input vector $\mathbf{u}$ and output vector $\mathbf{y}$. However, any invertible full-rank transformation matrix can be used to map **x** to an alternative state vector realisation containing linear combinations of **x**. Two useful transformations are "eigenmode" and "balanced realisation". The simpler one is the eigenmode transformation, which as described in Section 2.3 transforms the **A** matrix into a diagonal matrix of eigenvalues. These eigenvalues are actually the $1/\tau_v$ vessel time constants of the eigenmodes, and it is possible to reduce the model by sorting the eigenmodes in order of time constant and deleting the most rapidly decaying modes that would disappear in any timescale of interest.

However, trial and error has found that this truncation can sometimes lead to numerical instability in the final calculation, and this is not the most optimal way of reducing the number of elements in the state vector. In control theory, there is a more elegant way of optimising the state vector, often referred to as balanced truncation. The system is first transformed to a balanced realisation [5], by mapping to an alternative state vector in which each state variable has matching controllability and observability grammians. Essentially this means that each state has the same sensitivity to the input dynamics as it has influence on the output dynamics, in a time-integral sense. By ranking the state variables in grammian order, the first ones contain the strongest essence of the input to output transfer function, whilst the last ones have almost no contribution to the input–output transfer function and can be deleted. Experimentation has shown that the number of states needed to represent a 78-element passive circuit model can be reduced to 24 with very little impact on the observed solution of predicted vessel current.

2.3. Transformation to eigenmode representation

As introduced above, an eigenmode representation can be useful for computational efficiency as well as model reduction, due to the production of a diagonal matrix. In detail the procedure is as follows: we begin with the eigenvalue decomposition of **A** from Eq. (5):

$$\mathbf{A}\cdot\mathbf{W}=\mathbf{W}\cdot\mathbf{\Lambda} \tag{7}$$

where **W** is a full matrix whose columns are eigenvectors of **A**, and **Λ** is the diagonal matrix of eigenvalues, in this case the *R*/*L* inverse time-constants of the vessel.To transform **x** to $\mathbf{x}_e$, a state vector of eigenmodes, we substitute:

$$\mathbf{x}=\mathbf{W}\cdot\mathbf{x}_e \tag{8}$$

into Eq. (5) and use Eq. (7) to get:

$$\frac{d\mathbf{x}_e}{dt}=\mathbf{\Lambda}\cdot\mathbf{x}_e+\mathbf{W}^{-1}\cdot\mathbf{B}\cdot\mathbf{u} \tag{9}$$

$$\mathbf{y}=\mathbf{C}\cdot\mathbf{W}\cdot\mathbf{x}_e+\mathbf{D}\cdot\mathbf{u} \tag{10}$$

Now instead of having to calculate the full matrix-vector product **A**·**x**, we can just take the element-wise product of the main diagonal of **Λ** with $\mathbf{x}_e$. Note that even the reduced system resulting from the balanced truncation method of model reduction described in Section 2.2 can still be transformed to eigenmodes to benefit from the faster $\mathbf{\Lambda}\cdot\mathbf{x}_e$ calculation.

2.4. Discretisation of the ODE

The off-line code applies a continuous Runge-Kutta solver to the ODE, but this is unnecessary since the data is only available at discrete sample times. When the ODE solver calls for the value of d**x**/d*t*(**x**,*t*) at a minor integration step between sample times, the model function in the off-line code simply returns a linear sum of the values already obtained at the preceding and following sample times. Therefore the fixed ratios used by the ODE solver to combine these minor integration steps could have been folded into the matrix coefficients to allow the ODE to be solved with a single-step calculation at each sample time only. The proper way to do this is discretisation, a process where the input data is assumed to be constant (zero-order hold) or linearly interpolated (first-order hold) between the sample times and, given a fixed sample time, the analytic solution of the ODE is expressed by transforming the coefficients so that only a simple calculation at each time step is required. In MATLAB the c2d function with the 'first order hold' option can do this automatically, discretising the continuous state-space representation (5) and (6) above to the discrete representation:

$$\mathbf{x}(n+1)=\mathbf{A}_d\cdot\mathbf{x}(n)+\mathbf{B}_d\cdot\mathbf{u}(n) \tag{11}$$

$$\mathbf{y}(n)=\mathbf{C}_d\cdot\mathbf{x}(n)+\mathbf{B}_d\cdot\mathbf{u}(n) \tag{12}$$

where *n* is the sample number.

Without going into the detail, the discretisation process [6] requires the matrix exponent of **A**·d*t*, where d*t* is the sample interval. This is another case where the eigenmode representation is useful, because the matrix exponent of **Λ**·d*t* is simply found by taking the scalar exponent of each main diagonal element.

2.5. Runtime code optimisation

The control code runs on a PowerPC 7400 processor, which includes a set of 128-bit vector registers and an AltiVec^{TM} vector engine. These allow floating point calculations on four consecutive 32-bit values in a single operation. The size of the state vector was deliberately chosen to be a multiple of 4 for this reason, so that the algorithm could be implemented using the AltiVec^{TM} vector extensions to the C language [7]. The floating-point unit in the processor has a four-stage pipeline. This means that when a floating-point operation is performed, its result will be unavailable for four instruction cycles, so if the following instruction depends on this result, the CPU will stall waiting for the operation to complete. To make the CPU run efficiently it is therefore useful to unroll loops by four, so that each instruction is expanded to four instructions on four consecutive values. Because in this case each value is a 128-bit quad-float, each iteration of the loop actually processes 16 float values at a time, so the code implementation has to include the handling of data sets that are not multiples of 16. An additional optimisation provided by the AltiVec^{TM} unit is the ability to provide compiler hints to preload the cache memory with the coefficient data from **A**, **B**, **C** and **D** in a timely manner, so that when they are called upon the data does not have to be read from main memory.

2.6. Scheduling for low latency

In the discrete representation, the calculations can be separated into the part that can be done before the input data for a cycle is available and the part that depends on the input data. To determine $\mathbf{y}(n)$ for the nth control cycle, the slack period at the end of the $(n-1)$ cycle can be used to pre-calculate the $\mathbf{C}\cdot\mathbf{x}(n)$ component, since $\mathbf{x}(n)$ is only a function of $\mathbf{x}(n-1)$ and $\mathbf{u}(n-1)$, which are already available. Then only $\mathbf{D}\cdot\mathbf{u}(n)$ needs to be calculated in the time-critical phase after the measurement value $\mathbf{u}(n)$ is obtained, and added to the pre-calculated $\mathbf{C}\cdot\mathbf{x}(n)$ in order to provide the result $\mathbf{y}(n)$, i.e. the passive current data to be used by rtefit and the control algorithm. Likewise, after the controller outputs for cycle n have been sent, the value of $\mathbf{C}\cdot\mathbf{x}(n+1)$ can be pre-calculated in the slack period whilst waiting for the $\mathbf{u}(n+1)$ sample data, and so on. This approach minimises the contribution of the passive current estimation algorithm to the actual input–output latency.

3. Implementation issues and results

The EFIT passive structure model consists of 78 elements. After applying the variable substitution described in Section 2.1 and the model reduction by balanced truncation as described in Section 2.2, it was possible to achieve good representation of this system with just 24 states. However, when attempting to transform the system to the computationally more efficient eigenmode representation as described in Section 2.3, a side effect of this model reduction process was discovered: the 24-state model was found to have complex eigenvalues. It is not possible for the original model to have complex eigenvalues because they are the (real) inverse time constants of the passive system, so this must be an artefact of the model reduction process. Since the computational overhead of handling complex numbers would be much more than with real numbers, the model reduction was repeated with less aggressive truncation, and the 28-state model was found to have real values. The number of states was kept to a multiple of 4 to permit the use of the AltiVec$^{\mathrm{TM}}$ optimisations described in Section 2.5.

The model was discretised as described in Section 2.4 and implemented in the C language as a real-time algorithm in PCS, but split into the “me-critical” and “next cycle pre-calculation” stages as described in Section 2.6. The total extra computation overhead was found to be 145 μs per 500 μs control cycle. The AltiVec$^{\mathrm{TM}}$ optimisations described in Section 2.5 were then applied, which reduced the total overhead to 65 μs, with the time-critical part taking just 27.5 μs.

When the results of the EFIT pre-processor were first compared with the real-time version, some considerable disagreement was observed in the case of plasma shots. After careful analysis it was established that actually the problem was due to the way the EFIT pre-processor did the software differentiation of plasma current with an interpolated timebase, leading to poor results. This was corrected by implementing the change of variable method described in Section 2.1 for the standard EFIT pre-processor, which led to improvements in EFIT reconstructions as a by-product of this development work for real-time implementation.

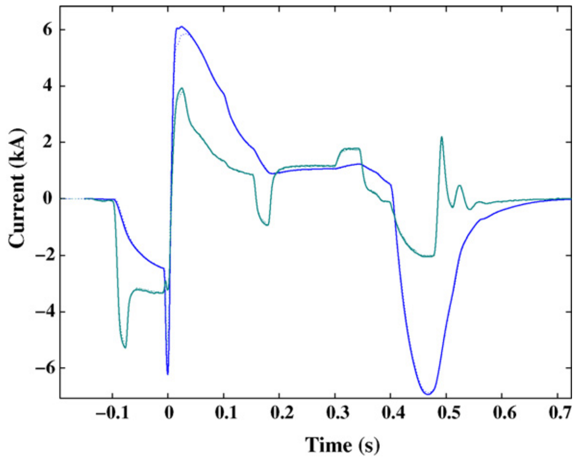

Fig. 2. Overlay comparison of a selection of passive element currents according to the EFIT pre-processor model (solid lines) and according to the real-time version (dotted lines).

Fig. 2 shows the predicted time evolution of currents in a selection of the passive elements, according to the EFIT pre-processor code (solid lines) and the real-time version (dotted lines). Both of these now use the auxiliary variable in the ODE to avoid problems caused by software differentiation. In most cases the two sets of traces overlay nearly perfectly so that only one trace is visible for each element for most of the duration of the shot.

With these encouraging results, a preliminary attempt was made at radial position control using rtefit, with successful results. Fig. 3 shows good agreement between EFIT and rtefit reconstructions of shot 16427. Fig. 4 shows how the edge radius was held at the reference value from shortly after the

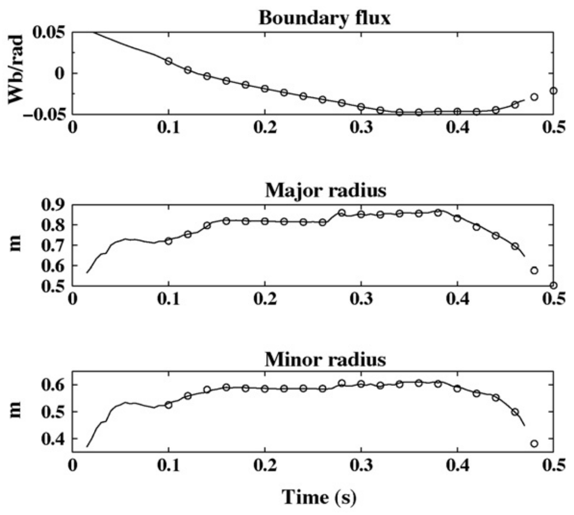

Fig. 3. Comparison of the computed boundary flux, major and minor plasma radii according to EFIT (solid lines) and rtefit (circles). Note that the rtefit data is from post-processing of data arrays stored at 20 ms intervals. The data available in real-time for control is expressed in terms of flux at the target boundary position, not in geometric units.

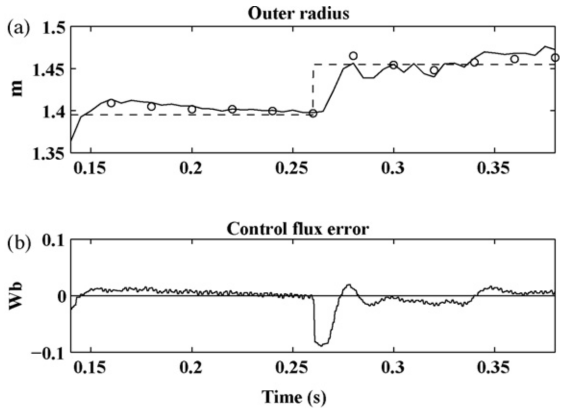

Fig. 4. First test of real-time feedback control on outer radius using rtefit. (a) The outer radius, according to EFIT (solid line) and rtefit (circles), is seen to track the reference radius (dashed line), including satisfactory response to a step change in the target at 0.26 s. (b) The flux error at the target position of the plasma edge, showing that the tracking error visible in (a) is also seen by rtefit, i.e. the error is mainly from the controller, not rtefit.

start of rtefit feedback control at 140 ms, including a satisfactorily stable response to the step change in target radius at 260 ms.

4. Conclusions

The EFIT code on MAST needs to take account of passive currents in the thick conducting vessel and other support structures. It does so using a code that runs after the shot to solve the induction equation of the MAST circuit model.

The recent implementation of rtefit, a real-time version of EFIT for real-time plasma boundary control, has required the same passive current estimation to be implemented as a real-time code.

A range of optimisations as presented here have been used with success to implement a fast real-time estimator, which in turn has led to the successful implementation of real-time feedback control of the plasma edge radius as determined by rtefit.

Some of the results of this optimisation have also been applied to the original code for EFIT to improve its performance.

Acknowledgements

This work was funded jointly by the United Kingdom Engineering and Physical Sciences Research Council and by the European Communities under the contract of Association between EURATOM and UKAEA. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] L.L. Lao, H. St. John, R.D. Stambaugh, A.G. Kellman, W. Pfeiffer, Reconstruction of current profile parameters and plasma shapes in tokamaks, Nucl. Fusion 25 (11) (1985) 1611–1622.
[2] J.R. Ferron, M.L. Walker, L.L. Lao, H.E.S.T. John, D.A. Humphreys, J.A. Leuer, Real time equilibrium reconstruction for tokamak discharge control, Nucl. Fusion 38 (7) (1998) 1055.
[3] B.G. Penaflor, J.R. Ferron, M.L. Walker, A structured architecture for advanced plasma control experiments, in: Proceedings of the 19th SOFT, Lisbon, Portugal, vol. 1, 1996, p. 965.
[4] G. McArdle, J. Storrs, J. Ferron, The MAST digital plasma control system, Fusion Eng. Des. 66–68 (2003) 761–765.
[5] B. Moore, Principal component analysis in linear systems: controllability, observability and model reduction, IEEE Trans. Autom. Control AC-26 (1981) 17–31.
[6] G.F. Franklin, J.D. Powell, M.L. Workman, Digital Control of Dynamic Systems, Second Edition, Addison-Wesley, 1990.
[7] AltiVec Technology Programming Interface Manual, ALTIVECPIM/D, Motorola, 1990.