Digital Filter Stepsize Control of DASPK and its Effect on Control Optimization Performance

by

Kirsten R. Meeker

M.S. Physics (California State University, Northridge) 1995
B.S. Physics (California State University, Northridge) 1988

A thesis submitted in partial satisfaction of the
requirements for the degree of
Master of Science

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, SANTA BARBARA

Committee in charge:
Professor Linda R. Petzold, Chair
Professor John R. Gilbert
Professor Mustafa Khammash

December 2004

The dissertation of Kirsten R. Meeker is approved:

Chair Date

Date

Date

University of California, Santa Barbara

September 2004

Digital Filter Stepsize Control of DASPK and its Effect on Control Optimization Performance

Abstract

Digital Filter Stepsize Control of DASPK and its Effect on Control Optimization Performance

by

Kirsten R. Meeker

Master of Science in Computer Science

University of California, Santa Barbara

Professor Linda R. Petzold, Chair

A new digital filter stepsize controller was implemented in the large-scale differential-algebraic equation solver DASPK3.1mod. The stepsize controller was taken from a modification to DASSL by Söderlind and Wang[21, 20], who investigated the effect of digital filter stepsize control on the computational stability of DASSL. We first tested the performance of DASPK3.1mod while solving several difficult problems, and then measured the effect of its improved computational stability on applications which use the differential-algebraic equation solver for sensitivity analysis and control optimization. We found that the new stepsize controller can substantially reduce the number of iterations needed by the optimizer. We conjecture that the performance improvement is due to the smoother behavior of the solution components with respect to perturbation of the problem parameters.

Professor Linda R. Petzold
Dissertation Committee Chair

To my parents, for passing on their sense of wonder and curiosity that enticed me to follow their path in science, and a sense of humor that kept me on it.

"A computer lets you make more mistakes faster than any invention in human history–with the possible exceptions of handguns and tequila." –Mitch Ratcliffe

Contents

List of Figures

List of Tables

Acknowledgments

I thank my committee for their time and dedication to educating students, my advisor, Linda Petzold, for her sage advice and thorough reviews, and Gustaf Söderlind and Lina Wang for their digital filter stepsize controller and code.

Chapter 1

Introduction

A new digital filter stepsize controller was implemented in the large-scale differential-algebraic equation (DAE) solver DASPK3.1mod[5]. This stepsize controller, taken from a modification to DASSL[17] (precursor to DASPK) by Söderlind and Wang[21, 20], was designed to produce a smoother stepsize control and output error of the DAE solver. Our goals were to test the new stepsize controller in DASPK3.1mod, and to find out whether this smoother output would improve the performance of control optimization codes that use DASPK. Control optimization codes are sensitive to the high frequency noise from the original stepsize controller in DASPK3.1, and we hypothesized that reducing the noise would allow the optimizer to converge more quickly.

The first step was to test the performance of the new stepsize controller in DASPK3.1mod while solving several difficult problems. The purpose of these tests was to ensure that the stepsize controller was producing smoother stepsize control, while preserving the accuracy of the solution of several different challenging problems. Then the effect of the DAE solver's improved computational stability on sensitivity analysis and control optimization was measured. We found that the new stepsize controller can substantially reduce the number of iterations needed by the optimizer. We conjecture that the performance improvement is due to the smoother behavior of the solution components with respect to perturbation of the problem. Step size control of ordinary differential equation (ODE) and

DAE solvers has typically used a simple linear controller, where the stepsize h is dependent on the local error estimate r_n. Limits are often imposed to prevent stepsize changes greater than a given factor, and cut-outs or dead-zones are used to prevent small stepsize changes. Figure 1.1 illustrates the stepsize control of DASPK3.1. Without limits or cut-outs it would show a straight line of slope $-1/k$, where $k = 1$ is the order of the stepsize controller.

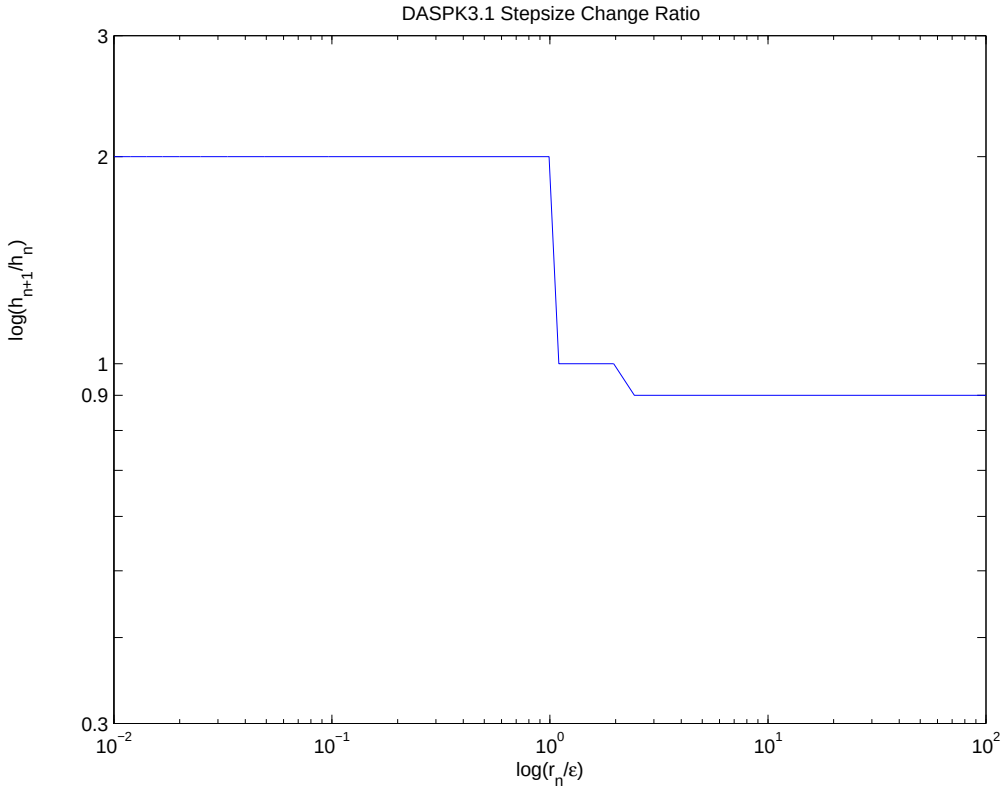


Figure 1.1: DASPK3.1 Stepsize Change Ratio. h_{n+1}/h_n is the ratio of 2 successive stepsizes, and r_n/ϵ is a fraction of the desired error tolerance. If the error is small, $r_n/\epsilon < 1$, the stepsize change is limited to doubling. If the error is large, $r_n/\epsilon > 2$, the stepsize is reduced to 90% of the previous stepsize. If the error is very small, $1 < r_n/\epsilon < 2$, there is a cut-out where the stepsize does not change.

Limits and cut-outs introduce periods of time where the control is disengaged. The effect is to create numerical "noise" that degrades the one-to-one correspondence between the requested tolerance and the achieved accuracy of the solution. The magnitude of this noise is highly variable, depending on small changes in

the computational setup. Söderlind and Wang [21] give the example of small changes in the roundoff of constants, that could be produced by changing compilers, causing very different amounts of work to achieve the same accuracy. They have proposed a new digital filter stepsize control that eliminates the stepsize discontinuities caused by limits and cut-outs and produces better tolerance proportionality.

Tolerance proportionality can be characterized by the relation

$$c \cdot TOL \leq \|e\| \leq C \cdot TOL, \tag{1.1}$$

where c and C are constants. Codes often behave more like

$$c \cdot TOL^{\alpha} \leq \|e\| \leq C \cdot TOL^{\alpha}, \tag{1.2}$$

which is computationally stable provided C/c is small, but is not tolerance proportional. Tolerance proportionality is a more restrictive condition than the requirement for computational stability.

The purpose of this investigation has been to determine whether better computational stability in DASPK [13] will improve the performance of design optimization and optimal control software. That software minimizes a nonlinear functional subject to the solution of a DAE system and potentially to other constraints. In several widely used approaches for solving the optimization problem, the DAE solver is used to compute both the objective functional and the derivative matrices (which are computed by sensitivity analysis in the DAE solver[13, 14]) for the optimization. Derivative-based optimizers which are normally employed for the optimization expect a function that depends smoothly on the optimization parameters. This is not generally the case for a function that must be computed by an ODE or DAE solver, because of the effects of adaptivity. One might expect that the smoother function resulting from a tolerance proportional DAE solver could lead to fewer iterations in the optimization.

We began by implementing the new stepsize controller in the DAE and sensitivity solver DASPK3.1. Then we evaluated the solutions, sensitivity derivatives,

and performance of DASPK3.1 and DASPK3.1mod (with the new stepsize control) for several initial value problems from the Bari University test set[15], to compare the stepsize control and to confirm previous results from Söderlind and Wang [21, 20] which used DASSL, a precursor to DASPK. Finally, we compared the optimization results and performance using both versions of DASPK3.1 for several optimal control test problems.

Chapter 2

Background

In this chapter, we describe the numerical methods used by DASPK and COOPT to solve optimal control problems, and the digital filter stepsize controller. DASPK methods that are described include the backward differentiation formula (BDF), the approach used to vary the order and stepsize, and how DASPK efficiently computes sensitivities. A brief history of the evolution of DASPK is given. Formulation of optimal control problems and a description of the multiple shooting method used in COOPT follows. Finally, the properties of the digital filter stepsize controller are compared with those of the original stepsize controller.

2.1 Simulation

In this subsection we briefly describe the methods and options available in the DAE solver DASPK3.1. DASPK3.1 uses the variable order, variable stepsize, backward differentiation formulas (BDF) [2] to compute the solution to the differential-algebraic (DAE) system

$$\begin{aligned} \mathbf{F}(t, \mathbf{y}, \mathbf{y}') &= 0 \\ \mathbf{y}(t_0) &= \mathbf{y}_0, \end{aligned} \tag{2.1}$$

where $\mathbf{y}$ is a vector of state variables and t represents time. Backward differentiation formula methods are so named because they approximate the derivative

using past values of the solution $\mathbf{y}$,

$$\mathbf{F}(t_n, y_n, \frac{1}{h\beta_0}\sum_{i=0}^{k}\alpha_i y_{n-i}) = 0, \tag{2.2}$$

where $\alpha_i, i = 0, \ldots, k$ and β_0 are coefficients of the BDF method. The order of the approximation can be varied by changing the number of past solution values used.

DASPK3.1 uses a modified Newton's method to solve the nonlinear system at each time step. The nonlinear system from (2.2) can be rewritten as a function whose root is the solution we are seeking,

$$g(y_n) = \mathbf{F}(t_n, y_n, \frac{1}{h\beta_0}\sum_{i=0}^{k}\alpha_i y_{n-i}) = 0. \tag{2.3}$$

Newton's method produces

$$\mathbf{y}_n^{\nu+1} = \mathbf{y}_n^{\nu} - (\frac{\partial g}{\partial \mathbf{y}})^{-1} g(\mathbf{y}_n^{\nu}) \quad \nu = 0, 1, \ldots. \tag{2.4}$$

The superscript ν is used to represent the iteration number. DASPK3.1 gives the user the option of using a direct method, or GMRES, a preconditioned Krylov iterative method, to solve the linear system at each Newton iteration. If the iterative method is chosen, the solution to the linear system is approximated to within a user-specified tolerance.

In the discussion of the BDF method so far, the order p and stepsize h have been constants. In all versions of DASSL and DASPK, the order is varied depending on the estimated stepsize possible at each order, and on whether successive terms of the approximating polynomial

$$|h^{p-1}\mathbf{y}^{(p-1)}|, |h^{p}\mathbf{y}^{(p)}|, |h^{p+1}\mathbf{y}^{(p+1)}|, |h^{p+2}\mathbf{y}^{(p+2)}| \tag{2.5}$$

are monotonically increasing or decreasing.

DASSL and DASPK use a variable stepsize to control both the local error and the efficiency of the code. Basically, the new stepsize $h_{n+1} = rh_n$ is selected so that the estimated error is a fraction of the user's desired error tolerance ETOL:

$$r = \left(\frac{fracETOL}{EST}\right)^{\frac{1}{p+1}}, \tag{2.6}$$

where EST is the estimate of the local truncation error and $\hat{p}$ is the new order. In addition to this basic stepsize control, limits on the amount the stepsize can increase are applied. Cut-outs, or periods of time when the stepsize is not changed, are used to prevent very small stepsize changes. Small stepsize changes are avoided because they may result in a need to re-evaluate the Jacobian (or the preconditioner in the case of iterative methods).

DASPK3.1 has evolved from DASSL[17] through several stages. DASSL was designed to use the fixed-leading-coefficient backward differentiation formulas (BDF) to solve index-1 DAEs. It uses Newton's method to solve for y_n in the BDF formula, and direct methods to solve the linear system at each Newton iteration. DASPK1.0[7] added a preconditioned Krylov iterative method, GMRES, for solving the linear systems. This method can be more efficient than the direct method for solving large-scale DAEs. DASPK2.0 added improved algorithms for the calculation of initial conditions for index-0 and index-1 systems. DASPK3.0 incorporated a sensitivity analysis capability, making use of automatic differentiation[3] to evaluate the sensitivity system, and improved calculation of initial conditions, as well as solution of Hessenberg index-2 DAE systems.

2.2 Sensitivity Analysis

Sensitivity analysis is useful in many engineering and scientific disciplines. Time histories of the sensitivities can be used to estimate parameters and to perform optimization, model reduction and experimental design. As mentioned previously, DASPK3.1 includes the capability to do sensitivity analysis.[13]

The sensitivities are the derivatives of the state variables $\mathbf{y}$ with respect to problem parameters $\mathbf{p}$. These derivatives can be computed by adding more equations to the system being solved. The number of additional equations is

$n_s = n_y \cdot n_p$[14]. Together, the new system is given by

$$\mathbf{F}(t, \mathbf{y}, \mathbf{y}', \mathbf{p}) = 0 \tag{2.7}$$

$$\frac{\partial \mathbf{F}}{\partial \mathbf{y}} s_i + \frac{\partial \mathbf{F}}{\partial \mathbf{y}'} s_i' + \frac{\partial \mathbf{F}}{\partial \mathbf{p}} = 0, \quad i = 1, \ldots, n_p \tag{2.8}$$

where $s_i = \frac{\partial \mathbf{y}}{\partial p_i}$.

The staggered corrector method available in DASPK3.1 solves the entire system in two steps[10]. First it computes the approximation to the solution $\mathbf{y}$ to (2.7) at the next time step using the BDF and Newton iteration. Then it solves the sensitivity equations (2.8) over the same time step using the BDF discretization and a second Newton iteration. The Jacobian matrix used to solve the original system and the sensitivity system are dependent only on the DAE solution, so they need only be computed once following the solution of the DAE on each time step.

2.3 Optimal Control

Given the differential-algebraic (DAE) system,

$$\mathbf{F}(t, \mathbf{y}, \mathbf{y}', \mathbf{p}, \mathbf{u}(t)) = 0, \tag{2.9}$$

$$\mathbf{y}(t_0, \mathbf{p}) = \mathbf{y}_0,$$

where p are parameters and $u(t)$ is the control function, the optimal control problem minimizes an objective function

$$\int \Psi(t, \mathbf{y}(t), \mathbf{p}, \mathbf{u}(t)) dt \tag{2.10}$$

subject to the solution of the DAE (2.9) and to some additional inequality constraints

$$G(t, \mathbf{y}(t), \mathbf{y'}(t), \mathbf{p}, \mathbf{u}(t)) \geq 0. \tag{2.11}$$

The software COOPT[23] developed by our research group uses the multiple shooting method to solve this problem. In the multiple shooting method, the time interval $[t_0, t_{max}]$ is divided into subintervals, and the DAE system determined

by 2.9 is solved in each subinterval $[t_i, t_{i+1}]$. Continuity of the solution across subintervals in ensured by introducing additional constraints to the optimization at the boundaries of the subintervals,

$$\mathbf{C}_1^i(\mathbf{Y}_{i+1}, \mathbf{Y}_i, p) \equiv \mathbf{Y}_{i+1} - \mathbf{y}(t_{i+1}, t_i, \mathbf{Y}_i, \mathbf{p}) = 0 \tag{2.12}$$

where $\mathbf{Y}_i$ are intermediate variables at the subinterval boundaries. At the beginning of each subinterval, the algebraic variables are perturbed to satisfy the original inequality constraints at the boundary. This introduces additional inequality constraints

$$\mathbf{C}_2^i(\mathbf{Y}_i, \mathbf{p}) \equiv g(t_i, \mathbf{Y}_i, \mathbf{p}) \geq 0. \tag{2.13}$$

The optimization problem in each interval is solved using SNOPT[12], an optimization code that uses sequential quadratic programming (SQP)[11]. SQP requires a gradient and Jacobian matrix containing the derivatives of the objective function and constraints with respect to the control parameters. These derivatives (sensitivities) are computed by DASPK3.1 using ADIFOR automatic differentiation software to generate the sensitivity equations[23]. Other methods using the shooting or multiple shooting method and a derivative-based optimizer operate similarly.

Another optimization solver that was used, KNITRO[8, 26], also uses SQP, but in addition uses trust regions to solve non-convex problems. This approach divides the original problem into subproblems divided by barriers. Each barrier subproblem is of the form

$$\min_{x,\, s} \quad f(x) - \mu \textstyle\sum_{i=1}^{m} ln s_i \tag{2.14}$$

$$\text{subject to} \qquad h(x) = 0 \tag{2.15}$$

$$g(x) + s = 0, \tag{2.16}$$

where $\mu > 0$ is the barrier parameter and s is a positive slack variable. As each subproblem is solved for the local minimum of the objective function, the barrier is adjusted to expand or shrink the trust region. Once all the subproblems have converged, the overall minimum can be found by comparing the solutions to the subproblems.

2.4 Digital Filter Stepsize Controller

In the first part of our work, we have replaced the original stepsize controller in DASPK3.1 by a new digital filter stepsize controller. The original elementary stepsize controller in DASPK3.1 is very effective, but its frequency response emphasizes high frequencies. As a consequence, the output stepsize h and resulting error $\hat{r}$ of the controller are rougher than the input disturbance $\hat{\phi}$. This makes the elementary stepsize controller suitable only for problems where smoothness of the computed solution is not a critical issue. Söderlind [19] has designed a stepsize low-pass filter, H211, with parameters that produce better frequency response while preserving stability. The H211 controller's more uniform frequency response makes it potentially more suitable for problems where smoothness is required.

The H211 digital filter stepsize controller satisfies the recursion relation

$$h_{n+1} = (\frac{\epsilon}{\hat{r}_n})^{\beta_1}(\frac{\epsilon}{\hat{r}_{n-1}})^{\beta_2}(\frac{h_n}{h_{n-1}})^{-\alpha_2}h_n, \tag{2.17}$$

where h_{n+1} is the stepsize at time $n+1$, h_n is the stepsize at time n, $\epsilon = fracETOL$, $\hat{r}_n$ is the local error estimate, $k\beta_1 = k\beta_2 = \alpha_2 = 1/4$, and $k = \hat{p} + 1$ where $\hat{p}$ is the order of convergence. The filter is named for what it is controlling, the step size h, and the *orders of dynamics, adaptivity*, and *filter*. The original stepsize controller in DASPK3.1 is H110 according to this naming scheme and satisfies the recursion relation

$$h_{n+1} = (\frac{\epsilon}{\hat{r}_n})^{\frac{1}{k}}h_n. \tag{2.18}$$

To better understand the terms used to characterize these controllers and compare their behavior, it is helpful to derive some equations describing the controllers' response to input disturbances.

The *stepsize* and *error transfer functions* are equations describing the stepsize and error response to controller input. They can be derived from the stepsize recursion relations (2.18) and (2.17), and the assumption that the local error is proportional to the step size

$$\hat{r}_n = \hat{\phi}_n h_n^k, \tag{2.19}$$

where $\hat{\phi}_n$ is the norm of the principal error function. Taking logarithms of (2.17-2.19) transforms them into the linear difference equations

$$logh_{n+1} = logh_n + \frac{1}{k}(log\epsilon - log\hat{r}_n) \tag{2.20}$$

$$logh_{n+1} = logh_n + \beta_1(log\epsilon - log\hat{r}_n) + \beta_2(log\epsilon - log\hat{r}_{n-1}) - \alpha_2(logh_n - logh_{n-1}) \tag{2.21}$$

$$log\hat{r} = klogh + log\hat{\phi}. \tag{2.22}$$

The first two equations can be simplified by introducing the forward shift operator q that advances the stepsize in time. Then q operating on $logh_n$ yields $logh_{n+1}$. Simplifying (2.20) yields

$$\begin{aligned} logh_{n+1} - logh_n &= \frac{1}{k}(log\epsilon - log\hat{r}_n) \\ (q-1)logh &= \frac{1}{k}(log\epsilon - log\hat{r}) \\ logh &= \frac{1}{k}\frac{1}{q-1}(log\epsilon - log\hat{r}). \end{aligned} \tag{2.23}$$

Simplifying (2.21) yields

$$\begin{aligned} logh_{n+1} - logh_n + \alpha_2(logh_n - logh_{n-1}) &= \beta_1(log\epsilon - log\hat{r}_n) + \beta_2(log\epsilon - log\hat{r}_{n-1}) \\ (q-1)(q+\alpha_2)logh &= (\beta_1 q + \beta_2)(log\epsilon - log\hat{r}_n) \\ logh &= \frac{\beta_1 q + \beta_2}{(q-1)(q+\alpha_2)}(log\epsilon - log\hat{r}_n). \end{aligned} \tag{2.24}$$

Figure 2.4 illustrates the interaction of the process and the controller represented by (2.22-2.24). The process transfer function $G(q) = k$ from (2.22), and the control transfer functions for the H110 and H211 controllers are

$$C(q) = \frac{1}{k}\frac{1}{q-1}, \quad C(q) = \frac{\beta_1 q + \beta_2}{(q-1)(q+\alpha_2)}, \tag{2.25}$$

from (2.23-2.24. To get a clearer picture of how the output stepsize and local error depend on the input $log\epsilon$ and $log\hat{\phi}$, we can solve for $log\hat{r}$ and $logh$, substituting

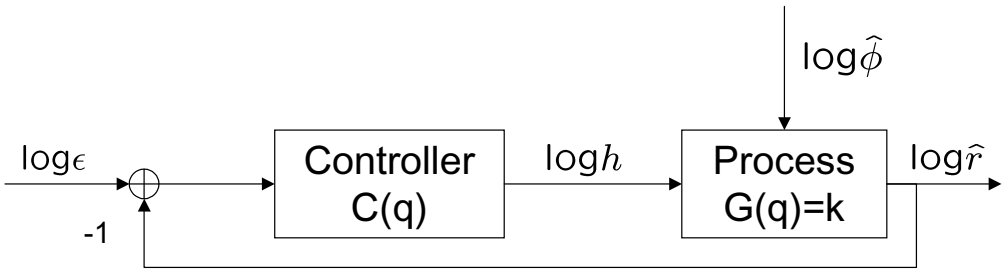


Figure 2.1: From Söderlind [19]. Adaptive stepsize feedback control system. The difference between the actual and desired error levels $log\epsilon - log\hat{r}_n$ is the input to the controller. The *control transfer function* $C(q)$ determines how that input is mapped to the output of the controller $logh$. Finally, the *process transfer function* $G(q)$ determines how $logh$, the stepsize, and $log\hat{\phi}$, the disturbance due to the ODE properties, is mapped to $log\hat{r}$, the local error estimate.

in $C(q)$ and $G(q)$, to obtain a system describing the closed loop dynamics

$$log\hat{r} = R_\epsilon(q)log\epsilon + R_{\hat{\phi}}(q)log\hat{\phi} \tag{2.26}$$

$$logh = H_\epsilon(q)log\epsilon + H_{\hat{\phi}}(q)log\hat{\phi}. \tag{2.27}$$

Since $log\epsilon$ is a constant, it can be set equal to zero leaving $H_{\hat{\phi}}(q)$, the stepsize transfer function, and $R_{\hat{\phi}}(q)$, the error transfer function. For $G(q) = k$ and a control function $C(q)$ these can be written as

$$H_{\hat{\phi}}(q) = -\frac{C(q)}{1 + k{\cdot}C(q)},\ R_{\hat{\phi}}(q) = \frac{1}{1 + k{\cdot}C(q)}. \tag{2.28}$$

$$\tag{2.29}$$

Substituting the control functions $C(q)$ for the H110 elementary controller and the H211 digital filter gives the stepsize and error transfer functions for each controller:

$$H_{\hat{\phi}}(q) = -\frac{1}{kq}, \qquad R_{\hat{\phi}}(q) = \frac{q-1}{q} \tag{2.30}$$

$$H_{\hat{\phi}}(q) = -\frac{\beta_1 q + \beta_2}{(q-1)(q+\alpha_2) + k(\beta_1 q + \beta_2)}, \quad R_{\hat{\phi}}(q) = \frac{(q-1)(q+\alpha_2)}{(q-1)(q+\alpha_2) + k(\beta_1 q + \beta_2)}. \tag{2.31}$$

Now we can define the terms used to name the controllers and discuss their significance.

Definition 2.4.1 *The order of dynamics p_D of the closed loop system is defined to be the degree of q in the denominator of the stepsize transfer function $H_{\hat{\phi}}(q)$.*

Definition 2.4.2 *Letting the error transfer function $R_{\hat{\phi}}(q)$ have all its poles strictly inside the unit circle, if $|R_{\hat{\phi}}(q)| = O(|q-1|^{p_A})$ as $q \to 1$, the order of adaptivity p_A of the controller is defined to be p_A.*

Definition 2.4.3 *Letting the stepsize transfer function $H_{\hat{\phi}}(q)$ have all its poles strictly inside the unit circle, if $|H_{\hat{\phi}}(q)| = O(|q+1|^{p_F})$ as $q \to -1$, the step size filter order at $q = -1$ is p_F.*

As mentioned at the beginning of this section, the elementary controller H110 and the digital filter H211 are named for these three terms. It can be seen that the controllers differ in order of dynamics and filter order, but have the same order of adaptivity.

A controller is stable if all poles (roots of the denominator of $H_{\hat{\phi}}(q)$) are strictly inside the unit circle. The single pole $q = 0$ of the elementary controller's stepsize transfer function $H_{\hat{\phi}}(q)$ is located at the origin, so the controller is highly stable. The order of dynamics quantifies how quickly and with what function the stepsize response decays to zero from an initial value, given no input disturbance (the homogeneous solution). The elementary controller has dynamic order one, so it decays linearly to a stepsize dependent on the initial value. The filter order determines the order of repeated averaging which smooths the stepsize. This elementary controller has filter order zero. Therefore it does no smoothing of successive stepsizes. Given that the controller is stable, has a linear decay function, and a reasonably fast response, the next property to examine is the controller's frequency response in both the stepsize and error outputs.

The stepsize frequency response $|H_{\hat{\phi}}(e^{i\omega})| \equiv 1$ for $\omega \in [0, \pi]$, so it is independent of the frequency ω. The output stepsize h will have the same spectral content as the input disturbance $\hat{\phi}$ because all frequencies have equal weighting. The error frequency response of a stable controller is $|R_{\hat{\phi}}(e^{i\omega})| = O(\omega^{p_a})$ as $\omega \to 0$. The

output error $\hat{r}$ will emphasize high frequencies of the input disturbance $\hat{\phi}$. The combined result is the undesired emphasis on high frequencies, making the output stepsize and error rougher than the input disturbance, and prompting the design of a digital filter stepsize controller.

The digital filter stepsize controller has a higher order of dynamics $p_D = 2$ and a higher filter order $p_F = 1$. The effect of these improvements is to give it a faster and smoother response. The order of adaptivity is unchanged from the elementary controller, so the error frequency response again will emphasize high frequencies of the input disturbance. However, the stepsize controller has been designed to shape the frequency response to be more uniform across all frequencies. This is done by placing the zeros (roots of the numerator of the stepsize transfer function) at frequencies we desire to suppress.

Söderlind and Wang demonstrate [21] that cut-outs in the DASSL step size control create significant variation in achieved accuracy versus tolerance, and in the number of function calls versus the achieved accuracy. Their results on the Chemakzo problem [4] show accuracies up to 1,000 times better than the requested tolerance and corresponding numbers of function calls up to 100 times greater than the number required to achieve accuracy matching the requested tolerance. It is important to note that DASSL produced an accuracy greater than or equal to the requested tolerance, so improvements in the step size control should not be expected to improve accuracy of the solution, but can be expected to reduce the amount of work done at some tolerances to achieve the required accuracy.

Chapter 3

Simulation Results

As mentioned earlier, we implemented the new digital filter stepsize controller in DASPK3.0. In this chapter we compare the performance of DASPK3.0 and DASPK3.0mod on several initial value problems. The test set is a collection of challenging initial value ODE problems collected by Bari University and distributed in the form of a report[15], a website[1], and a Fortran code. The report describes the problems, results for several ODE solvers including DASSL, and the Fortran code for both the problems and the solver drivers. Some modification to the DASSL driver was necessary to create a driver for DASPK3.0, but using the test problems was a very efficient way to test the new solver. The majority of coding was already done, and the problems and their solutions are well documented. Comparisons of the problem solutions and stepsizes chosen by DASPK3.0 and DASPK3.0mod were made for four problems from the test set: Chemical Akzo Nobel, High Irradiance Response (HIRES), Pollution, and Medical Akzo Nobel.

Problem	**RTOL/ ATOL**	**Significant Correct Digits**	**Integration Steps**	**Function Evaluations**	**Jacobian Evaluations**	**CPU Time (s)**
Chemakzo	1e-10/1e-10	8.78/10.00*	321/522	745/649	38/38	0.01/0.02
Medakzo	1e-7/1e-7	5.47/5.36	736/1375	1644/1681	65/62	0.81/1.04
Pollution	1e-10/1e-10	8.79/9.02	247/536	552/669	32/38	0.03/0.05
HIRES	1e-10/1e-10	8.42/8.95	575/905	1415/2233	51/46	0.03/0.04

*DASPK3.0mod/DASPK3.0, 2 GHz Intel Pentium 4, 260MB RAM, Compaq Visual Pro Fortran Version 6.1.0

Table 3.1: Comparison of Solver Performance on DAE Problems

From the results shown in Table 3.1, it can be seen that the new stepsize

controller reduces the number of integration steps and correspondingly the execution time, while preserving the accuracy as measured by the number of significant correct digits. It should be mentioned that we did not implement the tolerance rescaling used in DASSL by Söderlind and Wang, because it would not allow a like comparison between DASPK3.0 and DASPK3.0mod. The tolerance rescaling adjusts the difference between the tolerance and the number of significant correct digits.

The remainder of this section contains descriptions of the problems, their numerical solutions using both DASPK3.0 and DASPK3.0mod and the stepsizes chosen by the two solvers. In all cases the stepsizes chosen by the new digital filter controller were both much smoother and almost always larger than the stepsize chosen by the DASPK3.0 stepsize controller.

The Chemical Akzo Nobel problem[4], from Akzo Nobel Central Research in Arnhem, the Netherlands, describes a chemical process in which two species are mixed while carbon dioxide is continuously added. It is a stiff system of 6 nonlinear ODE's. The chemical reactions are given by

$$\begin{aligned} 2FLB + \frac{1}{2}CO_2 &\xrightarrow{k_1} FLBT + H_2O \\ ZLA + FLB &\underset{k_2}{\overset{k_2/K}{\longleftrightarrow}} FLBT + FLB \\ FLB + 2FLB + CO_2 &\xrightarrow{k_3} LB + nitrate \\ FLB.ZHU + \frac{1}{2}CO_2 &\xrightarrow{k_4} ZLA + H_2O \\ FLB + FLB &\longleftrightarrow FLB.ZHU, \end{aligned}$$

where $k_1 = 18.7$, $k_2 = 0.58$, $k_3 = 0.09$, $k_4 = 0.42$, and $K = 34.4$. The concentrations FLB, $[CO_2]$, [FLBT], [FLB], [ZLA], [FLB.ZHU] are the state variables $y(1), \ldots, y(6)$. The solutions, significant correct digits (scd), and stepsize from both DASPK3.0 and DASPK3.0mod for initial condition
$y_0 = (0.437, 0.00123, 0, 0, 0, 0.367)^T$ and $RTOL = ATOL = 1e-10$ are shown in Table 3.2 at $t = 180$ minutes, and in Figures 3.1 and 3.2 for the time interval $[0, 180$ minutes$]$. The solutions are indistinguishable from each other in the plots, though there is a difference of one significant digit in the solution at the end of

the time interval, $t = 180$ minutes. The stepsize control is much smoother using the digital filter stepsize controller.

	DAPK3.1mod	**DASPK3.0**
y(1)	$0.1150794937298346E{+}000$	$0.1150794919653201E{+}000$
y(2)	$0.1203831470559661E{-}002$	$0.1203831471628333E{-}002$
y(3)	$0.1611562879129449E{+}000$	$0.1611562887905489E{+}000$
y(4)	$0.3656156500854225E{-}003$	$0.3656156409791002E{-}003$
y(5)	$0.1708010852600139E{-}001$	$0.1708010884776064E{-}001$
y(6)	$0.4873531486873073E{-}0002$	$0.4873531287586157E{-}002$
scd	8.78	10.00

Table 3.2: Chemakzo Problem Solution at Time 180 minutes. scd is the number of significant correct digits.

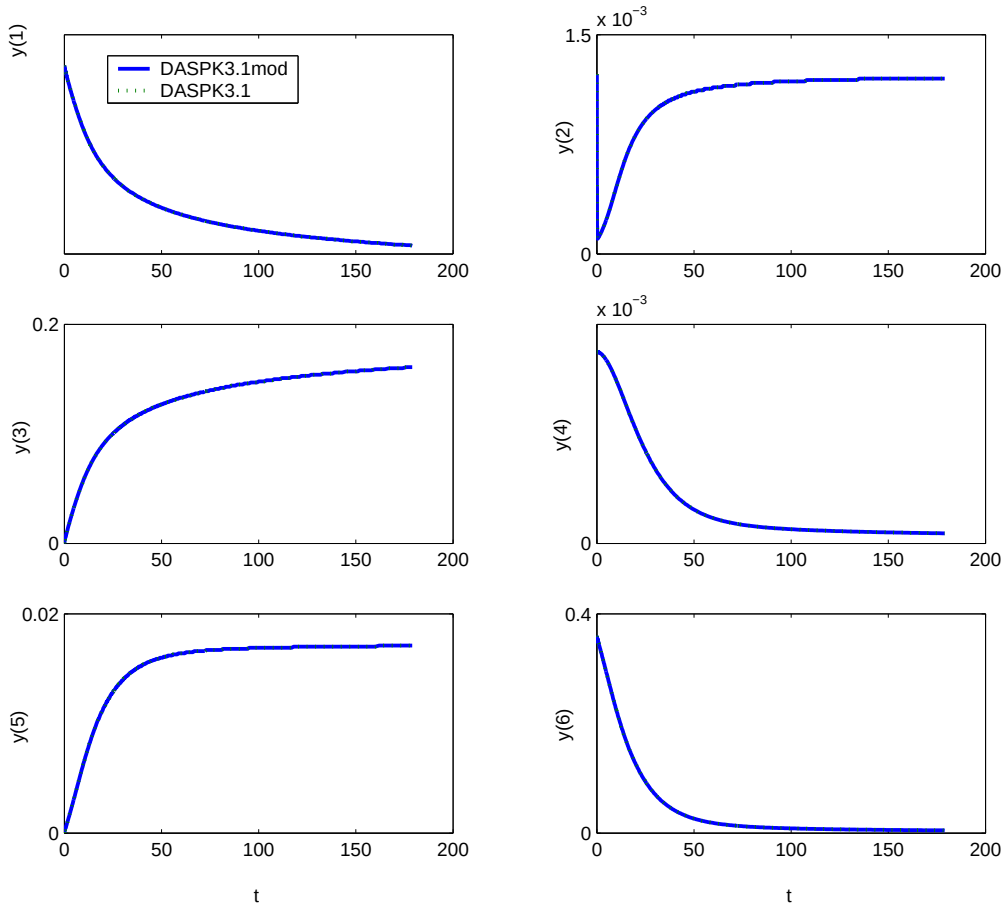


Figure 3.1: Chemakzo Problem Solution vs. Time

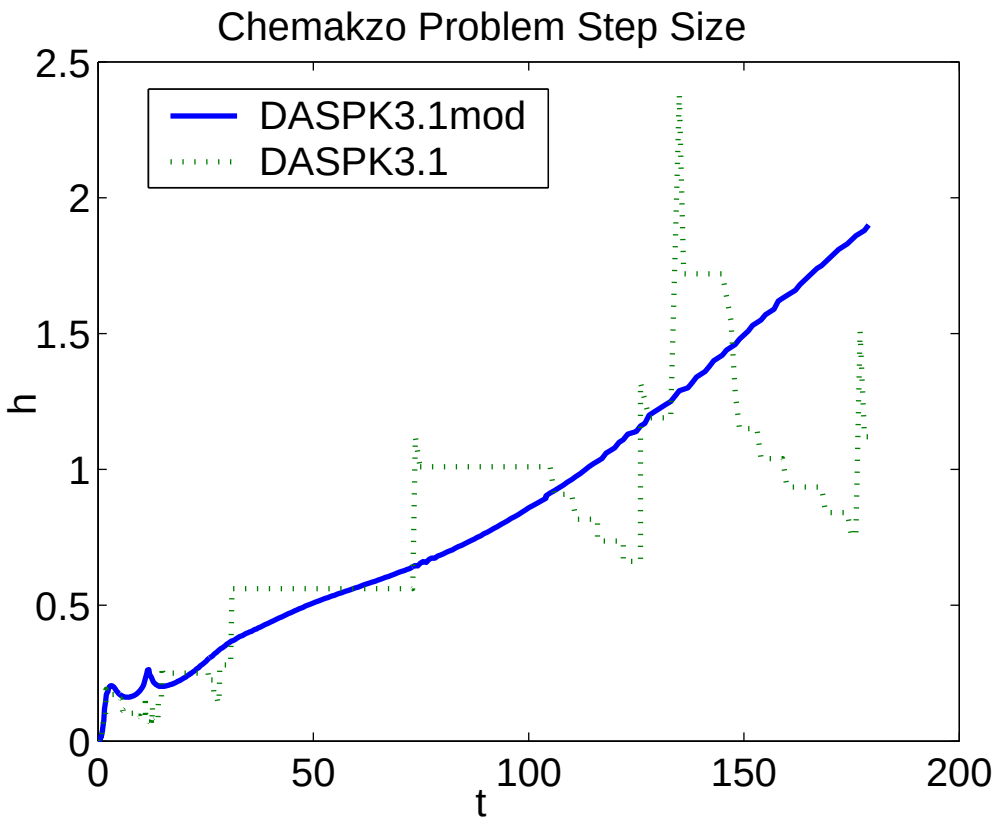


Figure 3.2: Chemakzo Problem Stepsize vs. Time

The Medical Akzo Nobel problem[24], from Akzo Nobel Central Research in Arnhem, the Netherlands, describes the penetration of a radio-labeled antibody into a tumor. The problem is a a reaction diffusion system in one spatial dimension consisting of two partial differential equations

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - kuv \tag{3.1}$$

$$\frac{\partial v}{\partial t} = -kuv, \tag{3.2}$$

where u is the concentration of the radio-labeled antibody, v is the tumor, x is the spatial dimension, and $k = 100$ is a constant. The initial and boundary conditions are given by

$$u(x,0) = 0,\ v(x,0) = v_0 \text{ for } x > 0, \tag{3.3}$$

$$u(0,t) = \phi(t) \text{ for } 0 < t < T, \tag{3.4}$$

where $v_0 = 1$ is a constant, and $T = 20$. The function ϕ is given by

$$\phi(t) = \begin{cases} 2 \text{ for } t \in (0,5]\,, \\ 0 \text{ for } t \in (5,20]\,. \end{cases} \tag{3.5}$$

This system is discretized to form the ODE system

$$\frac{dy}{dt} = f(t,y), \quad y(0) = (0, v_0, 0, v_0, \ldots, 0, v_0)^T, \tag{3.6}$$

where $y \in \Re^{2N}$, and $N = 200$. The function f is given by

$$f_{2j-1} = \alpha_j \frac{y_{2j+1} - y_{2j-3}}{2\Delta\varsigma} + \beta_j \frac{y_{2j-3} - y_{2j-1} + y_{2j+1}}{(\Delta\varsigma)^2} - k y_{2j-1} y_{2j}, \tag{3.7}$$

$$f_{2j} = -k y_{2j} y_{2j-1}, \tag{3.8}$$

where

$$\alpha_j = \frac{2(j\Delta\varsigma - 1)^3}{c^2}, \tag{3.9}$$

$$\beta_j = \frac{(j\Delta\varsigma - 1)^4}{c^2}. \tag{3.10}$$

$j = 1, \ldots, N, \Delta\varsigma = \frac{1}{N}, y_{-1}(t) = \phi(t), y_{2N+1} = y_{2N-1}$, and $c = 4$ is a constant. The solutions, significant correct digits (scd), and stepsize from both DASPK3.0 and

DASPK3.0mod for $RTOL = ATOL = 1e{-}7$ are shown in Table 3.3 and in Figures 3.3 and 3.4. The plotted solutions are indistinguishable, and the significant correct digits are identical at the end of the time interval. The digital filter stepsize is much smoother and larger in the second half of the time interval than the stepsize chosen by DASPK3.0.

	DAPK3.1mod	**DASPK3.0**
y(79)	$0.2341237711791777E{-}003$	$0.2339737545237306E{-}003$
y(80)	$0.4646450416982464E{-}064$	$0.1066553305170135E{-}021$
y(149)	$0.3597578098582422E{-}003$	$0.3595312994053164E{-}003$
y(150)	$0.1382729878954674E{-}044$	$0.7373226719740840E{-}027$
y(199)	$0.1174358772641044E{-}003$	$0.1173673938031241E{-}003$
y(200)	$0.6253945747668511E{-}005$	$0.6190791265794569E{-}005$
y(239)	$0.6863589782839472E{-}011$	$0.6855081012118170E{-}011$
y(240)	$0.9999997325839053E{+}000$	$0.9999997325881845E{+}000$
scd	5.47	5.36

Table 3.3: Medakzo Problem Solution at Time 20. scd is the number of significant correct digits.

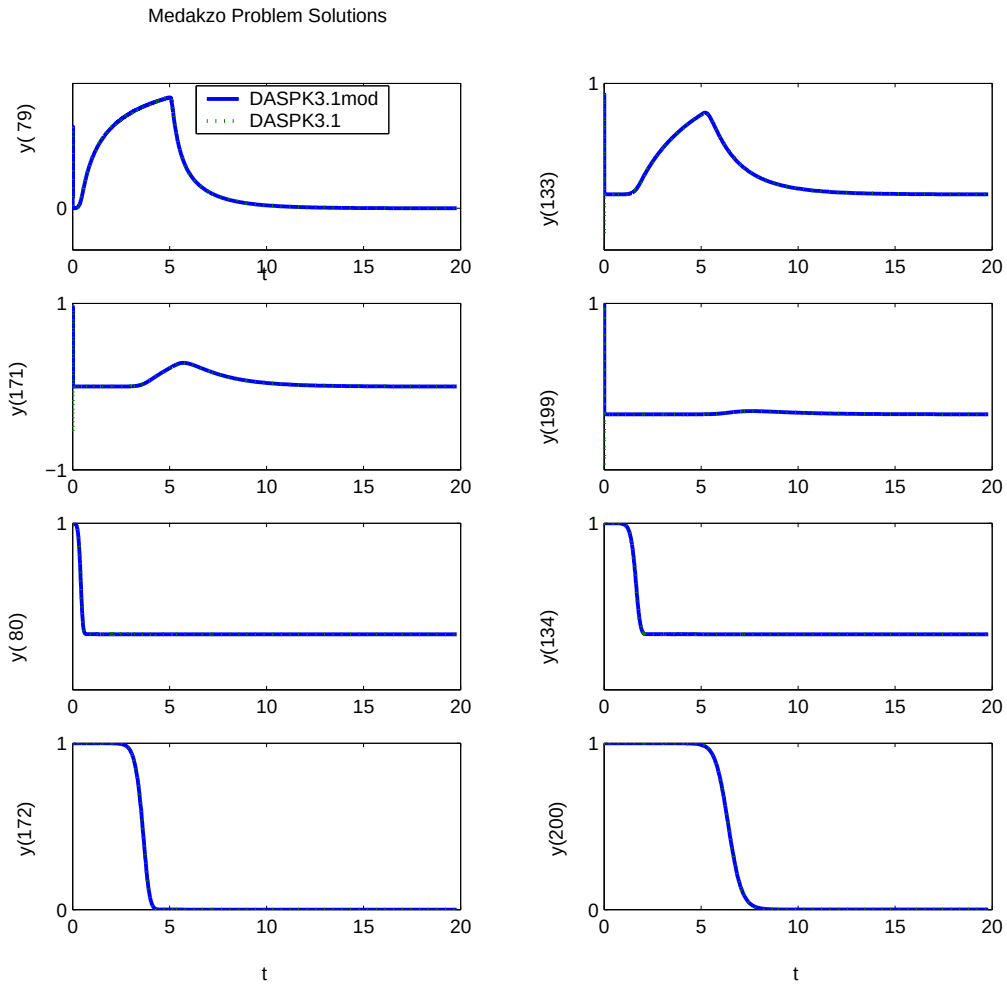


Figure 3.3: Medakzo Problem Solution vs. Time

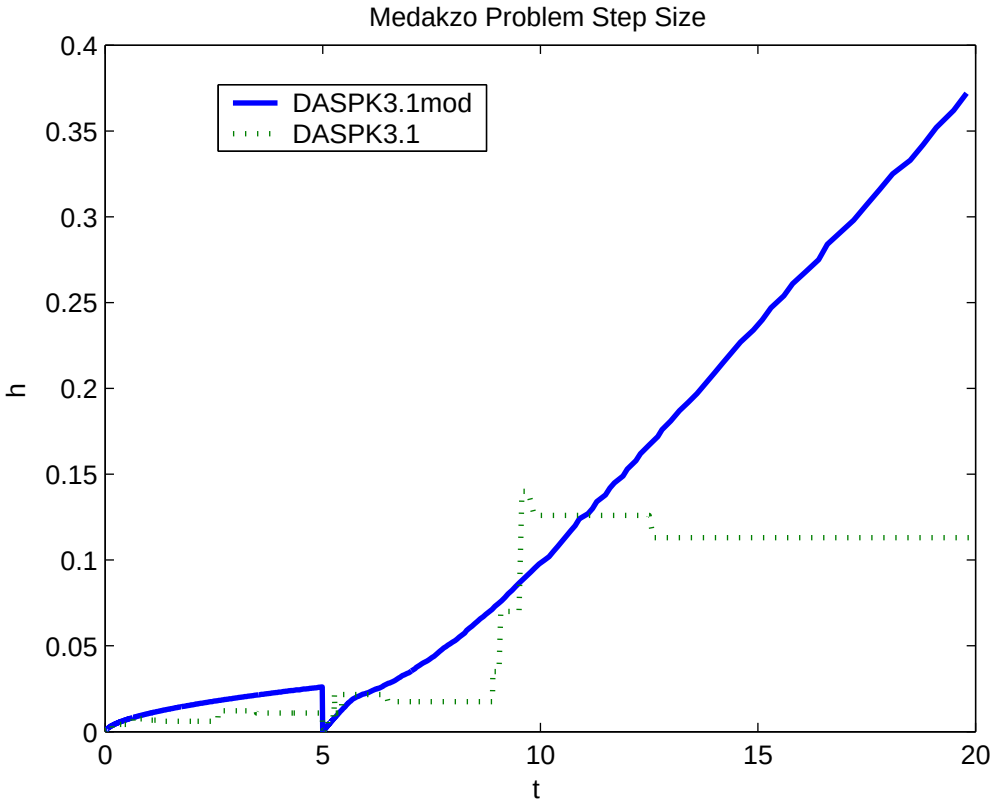


Figure 3.4: Medakzo Problem Stepsize vs. Time

The pollution problem describes the chemical reaction part of the air pollution model developed at The Dutch National Institute of Public Health and Environmental Protection[25]. It is a stiff system of 20 nonlinear ODEs. The chemical reactions are given by

$$
\begin{array}{lcllcl}
NO2 & \rightarrow & NO + O3P & NO2 + OH & \rightarrow & HNO3 \\
NO2 + O3 & \rightarrow & NO2 & O3P & \rightarrow & O3 \\
HO2 + NO & \rightarrow & NO2 + OH & O3 & \rightarrow & O1D \\
HCHO & \rightarrow & 2HO2 + CO & O3 & \rightarrow & O3P \\
HCHO & \rightarrow & CO & O1D & \rightarrow & 2OH \\
HCHO + OH & \rightarrow & HO2 + CO & O1D & \rightarrow & O3P \\
ALD & \rightarrow & MEO2 + HO2 + CO & SO2 + OH & \rightarrow & SO4 + HO2 \\
ALD + OH & \rightarrow & C2O3 & NO3 & \rightarrow & NO \\
C2O3 + NO & \rightarrow & NO2 + MEO2 + CO2 & NO3 & \rightarrow & NO2 + O3P \\
C2O3 + NO2 & \rightarrow & PAN & NO2 + O3 & \rightarrow & NO3 \\
PAN & \rightarrow & C2O3 + NO2 & NO3 + NO2 & \rightarrow & N2O5 \\
MEO2 + NO & \rightarrow & CH3O + NO2 & N2O5 & \rightarrow & NO3 + NO2 \\
CH3O & \rightarrow & HCHO + HO2. & & &
\end{array}
$$

The concentrations [NO2], [NO], [O3P], [O3], [HO2], [OH], [HCHO], [CO], [ALD], [MEO2], [C2O3], [CO2], [PAN], [CH3O], [HNO3], [O1D], [SO2], [SO4], [NO3], and [N2O5] are the state variables $y(1), \ldots, y(20)$. The solutions, significant correct digits (scd), and stepsize from both DASPK3.0 and DASPK3.0mod for initial condition $y_0 = (0, 0.2, 0, 0.04, 0, 0, 0.1, 0.3, 0.01, 0, 0, 0, 0, 0, 0, 0, 0.007, 0, 0, 0)^T$ and $RTOL = ATOL = 1e-10$ are shown in Table 3.4 at $t = 60$, and in Figures 3.5-3.7 for the time interval $[0, 60]$. As in the Medakzo problem results, the plotted solutions are indistinguishable, and the significant correct digits are nearly identical at the end of the time interval. The digital filter stepsize is much smoother and larger in the second half of the time interval than the stepsizes chosen by DASPK3.0.

	DAPK3.1mod	**DASPK3.0**
y(1)	$0.5646255316933575E{-}001$	$0.5646255384042490E{-}001$
y(2)	$0.1342484146780313E{+}000$	$0.1342484139617222E{+}000$
y(3)	$0.4139734211191470E{-}008$	$0.4139734260527444E{-}008$
y(4)	$0.5523139980065256E{-}002$	$0.5523140074876242E{-}002$
y(5)	$0.2018977314137636E{-}006$	$0.2018977323956053E{-}006$
y(6)	$0.1464541922911229E{-}006$	$0.1464541928232819E{-}006$
y(7)	$0.7784249219906239E{-}001$	$0.7784249112557803E{-}001$
y(8)	$0.3245075341152212E{+}000$	$0.3245075351205696E{+}000$
y(9)	$0.7494013603214714E{-}002$	$0.7494013410675476E{-}002$
y(10)	$0.1622293238858437E{-}007$	$0.1622293489370140E{-}007$
y(11)	$0.1135863891824073E{-}007$	$0.1135864079923440E{-}007$
y(12)	$0.2230505760992245E{-}002$	$0.2230505693568928E{-}002$
y(13)	$0.2087162835189467E{-}003$	$0.2087165428732562E{-}003$
y(14)	$0.1396921104684839E{-}004$	$0.1396921297030541E{-}004$
y(15)	$0.8964884867773863E{-}002$	$0.8964884649629724E{-}002$
y(16)	$0.4352846190098655E{-}017$	$0.4352846264820216E{-}017$
y(17)	$0.6899219701403960E{-}002$	$0.6899219699234935E{-}002$
y(18)	$0.1007802985960403E{-}003$	$0.1007803007650652E{-}003$
y(19)	$0.1772146395183316E{-}005$	$0.1772146436549738E{-}005$
y(20)	$0.5682942747253824E{-}004$	$0.5682942945684164E{-}004$
scd	8.79	9.02

Table 3.4: Pollution Problem Solution at Time 60. scd is the number of significant correct digits.

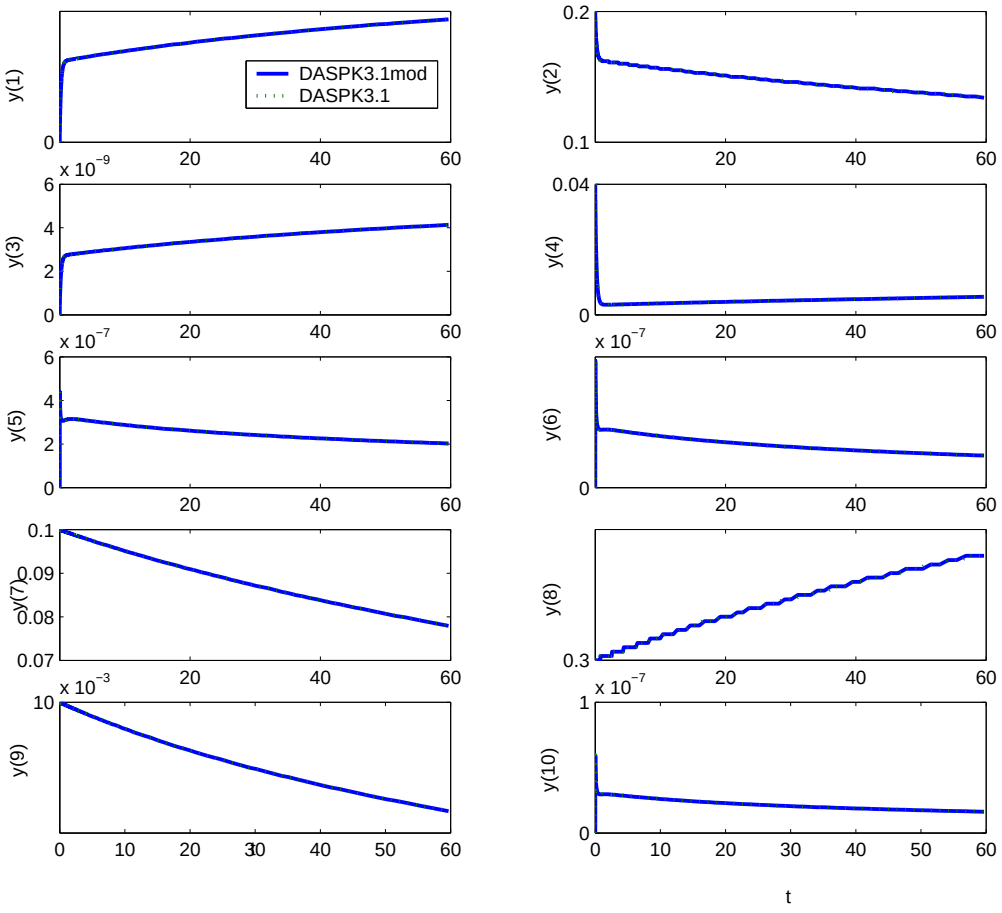


Figure 3.5: Pollution Problem Solution vs. Time, y(1),...,y(10)

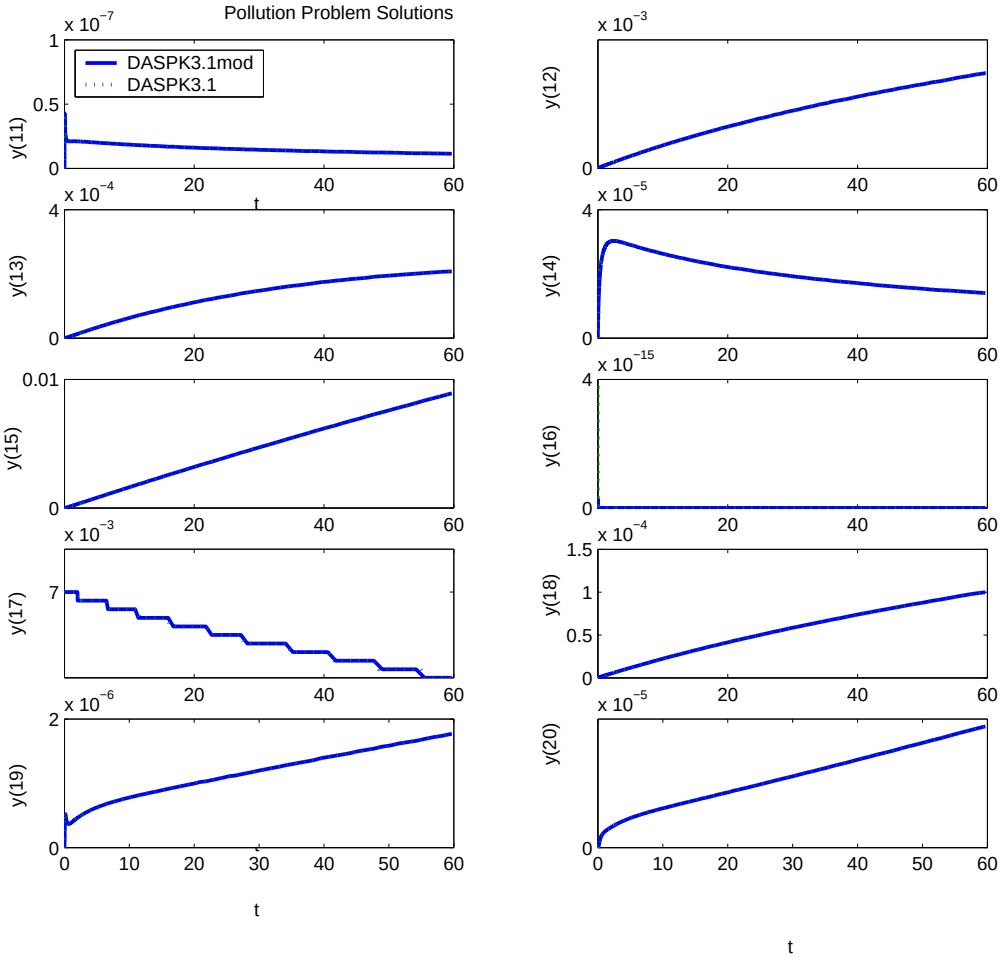


Figure 3.6: Pollution Problem Solution vs. Time, y(11),...,y(20)

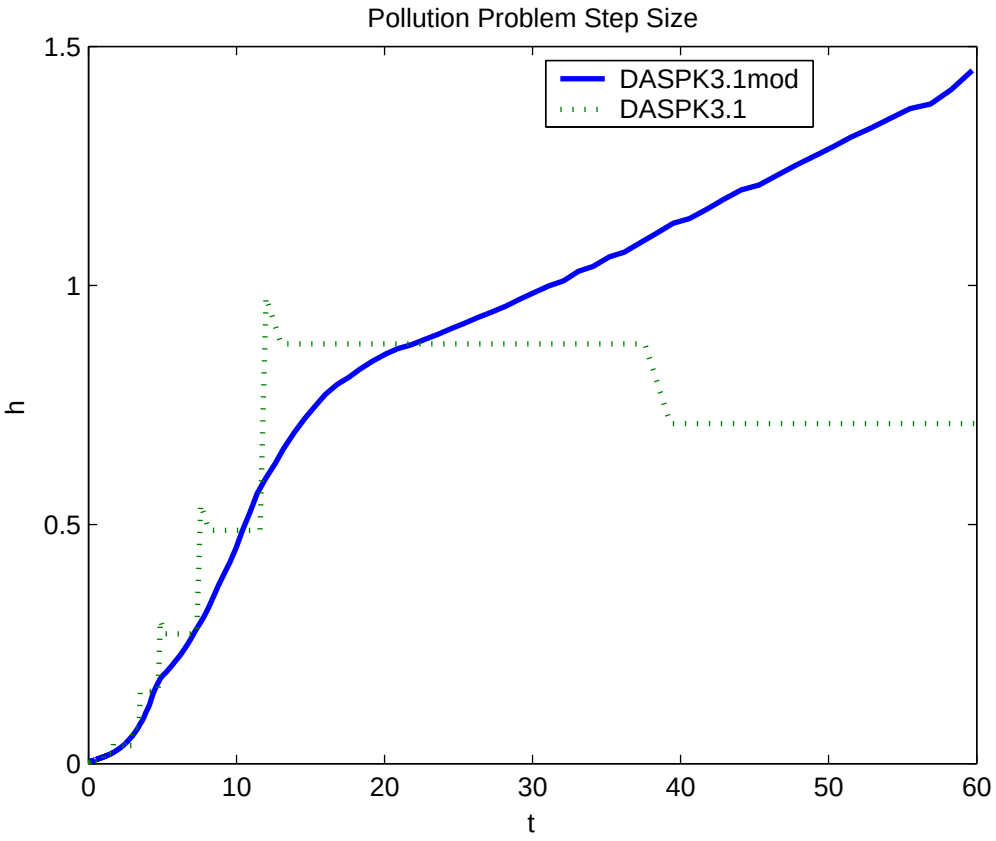
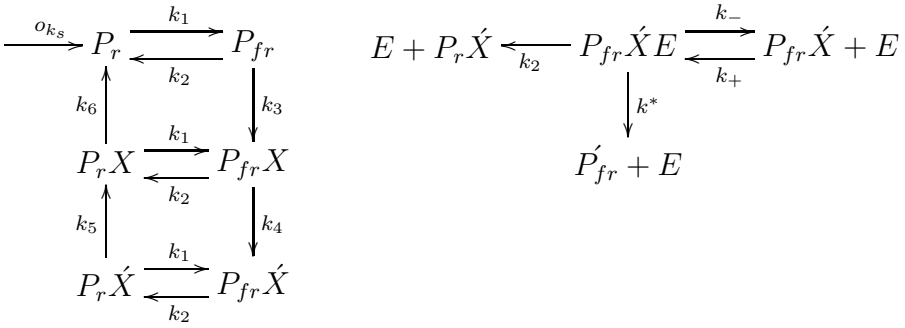


Figure 3.7: Pollution Problem Stepsize vs. Time

The High Irradiance Response (HIRES) problem[18] describes how light effects the photomorphogenesis of plants through phytochromes. The problem is a stiff system of 8 nonlinear ODEs. The chemical reactions are given by



(3.11)

where $k_1 = 1.71, k_2 = 0.43, k_3 = 8.32, k_4 = 0.69, k_5 = 0.035, k_6 = 8.32, k_+ = 280, k_- = 0.69, k^* = 0.69$, and $o_{k_s} = 0.0007$. P_r and P_{rr} are the red and far-red absorbing phytochromes. They can be bound by two receptors X and $\acute{X}$, and are partially influenced by enzyme E. The concentrations of $P_r, P_{fr}, P_rX, P_{fr}X, P_r\acute{X}$, $P_{fr}\acute{X}, P_{fr}\acute{X}E$, and E are the state variables $y(1), \ldots, y(8)$. The solutions, significant correct digits (scd), and stepsize from both DASPK3.0 and DASPK3.0mod for initial condition $y_0 = (1, 0, 0, 0, 0, 0, 0, 0.0057)^T$ and $RTOL = ATOL = 1e-10$ are shown in Table 3.5 at $t = 321.8122$, and in Figures 3.8 and 3.9 for the time interval $[0, 321.8122]$. Again, the plotted solutions are indistinguishable, and the significant correct digits are nearly identical at the end of the time interval. The stepsize control is much smoother using the new digital filter stepsize controller compared to the simple stepsize controller.

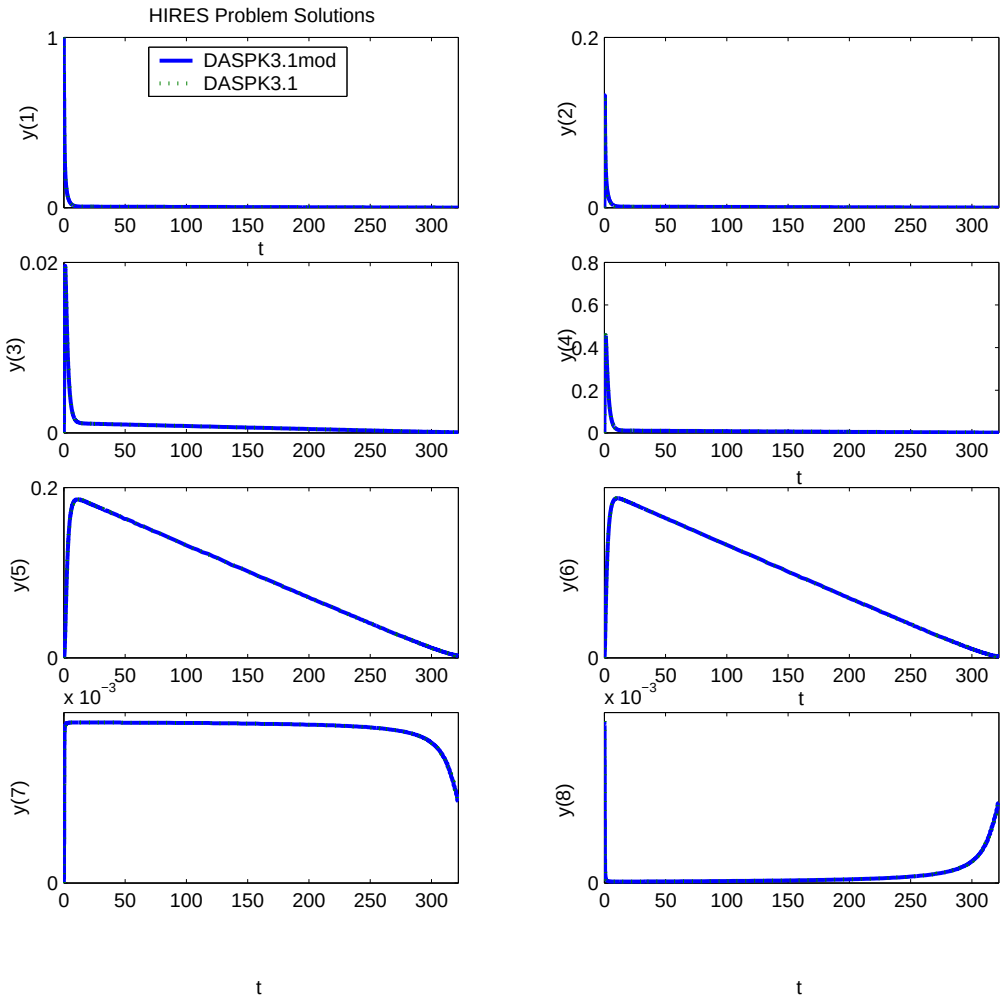


Figure 3.8: High Irradiance Response Problem Solution vs. Time

	DAPK3.1mod	DASPK3.0
y(1)	$0.7371312961567492E{-}003$	$0.7371312695033106E{-}003$
y(2)	$0.1442485802675641E{-}003$	$0.1442485750316180E{-}003$
y(3)	$0.5888730472692921E{-}004$	$0.5888729967037030E{-}004$
y(4)	$0.1175651414497739E{-}002$	$0.1175651366028672E{-}002$
y(5)	$0.2386357406760650E{-}002$	$0.2386356561712752E{-}002$
y(6)	$0.6238972088227086E{-}002$	$0.6238969366526370E{-}002$
y(7)	$0.2849999231945713E{-}002$	$0.2849998674827406E{-}002$
y(8)	$0.2850000768054287E{-}002$	$0.2850001325172629E{-}002$
scd	8.42	8.95

Table 3.5: HIRES Problem Solution at Time 321.8122. scd is the number of significant correct digits.

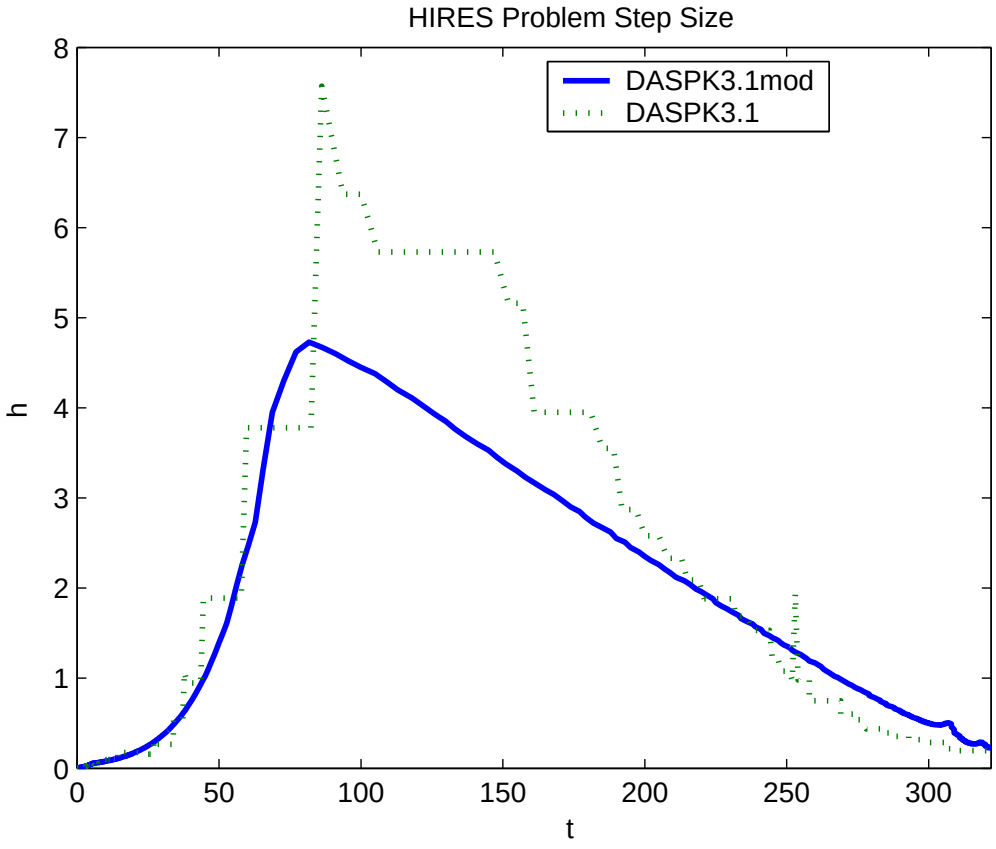


Figure 3.9: High Irradiance Response Problem Stepsize vs. Time

Comparison of the results of the solution of these IVP Test Set problems demonstrates that the digital filter stepsize control is much smoother that that of the original simple controller, and that there is no significant degradation in the accuracy of the solutions when using the new stepsize controller. The first order response (slope of stepsize plots) of both controllers is similar in all cases except when cut-outs were in effect during the second half of the Medakzo and pollution problem time intervals. The solutions during this period were nearly constant, and the digital stepsize controller appropriately increased the stepsize. In contrast, the simple controller decreased the stepsize then held it constant, indicating the cut-out was in effect.

Chapter 4

Sensitivity Analysis Results

This chapter contains the results of both DASPK3.0 and DASPK3.0mod's solutions of two sensitivity computation problems: the heat equation discretized on a square, and a predator-prey population modeled on a square. Table 4.1 shows the performance of the two solvers on these problems. In both cases the

Problem	RTOL/ ATOL	Max Solution Difference	Max Sensitivity Difference	Time Steps	Nonlinear Iterations	CPU Time (s)
Heat2D	1e-5	1e-5	1e-3	197/234	440/267	17.4/12.1
Food Web	1e-5	6.7e-5	1.4	109/128	204/152	1.0/1.0

*DASPK3.0mod/DASPK3.0

Table 4.1: Comparison of Solver Performance on Sensitivity Problems

new stepsize controller reduced the number of time steps taken, but increased the total number of nonlinear iterations and the execution time. The remainder of this section contains descriptions of the sensitivity problems, their numerical solutions using both DASPK3.0 and DASPK3.0mod, and the stepsizes chosen by the two solvers. Again, the stepsizes chosen by the new stepsize controller are almost always larger than those chosen by the original controller.

The first problem is a DAE system from the discretization of the heat equation

$$\frac{du}{dt} = p_1 \frac{\partial^2 u}{\partial x^2} + p_2 \frac{\partial^2 u}{\partial y^2}, \tag{4.1}$$

on a two dimensional unit square with zero Dirichlet boundary conditions. A uniform 42 x 42 mesh is used to discretize the partial differential equation (PDE)

(4.1) using a standard central finite difference approximation. The resulting 42 x 42 system of equations is comprised of ODEs for u at each interior point, and boundary conditions $u(x, y) = 0$ on the boundary. The initial conditions are $u(t = 0) = 16x(1 - x)y(1 - y)$, the parameters are $p_1 = p_2 = 1.0$ and the problem is solved over the time interval $t = [0, 10.24]$. The solutions differed by at most 10^{-5} and the sensitivities differed by at most 10^{-3}. Figure 4.1 shows the initial condition, solution, and sensitivity derivatives $\frac{\partial u}{\partial p_1}$ and $\frac{\partial u}{\partial p_2}$ at time 10.24. Figure 4.2 shows the stepsize versus time for both stepsize controllers.

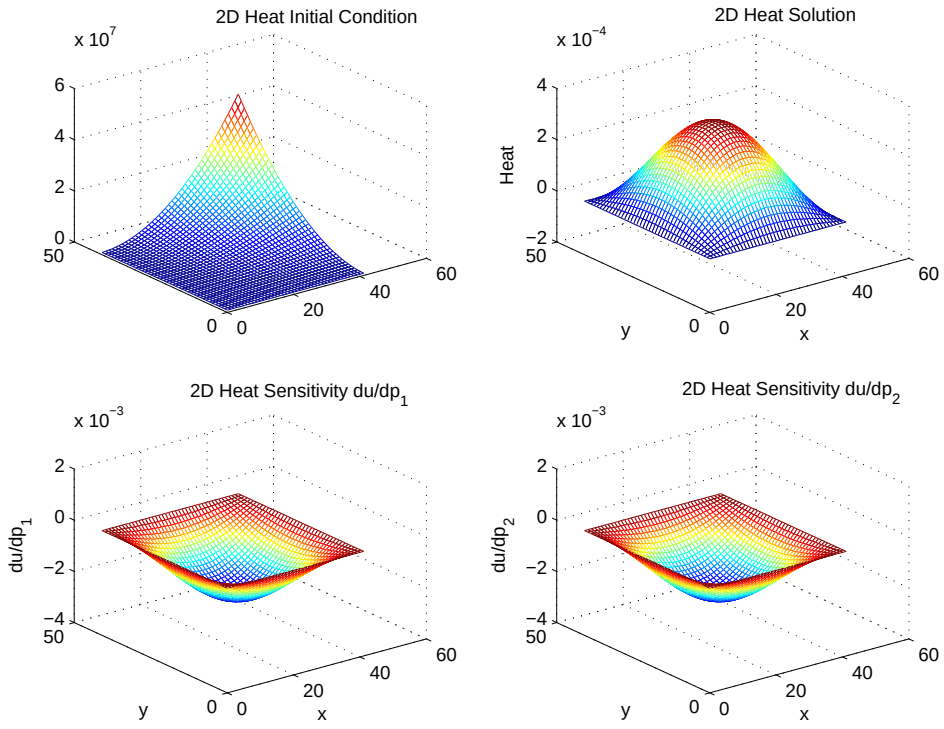


Figure 4.1: 2D Heat Problem Solution at Time 10.24

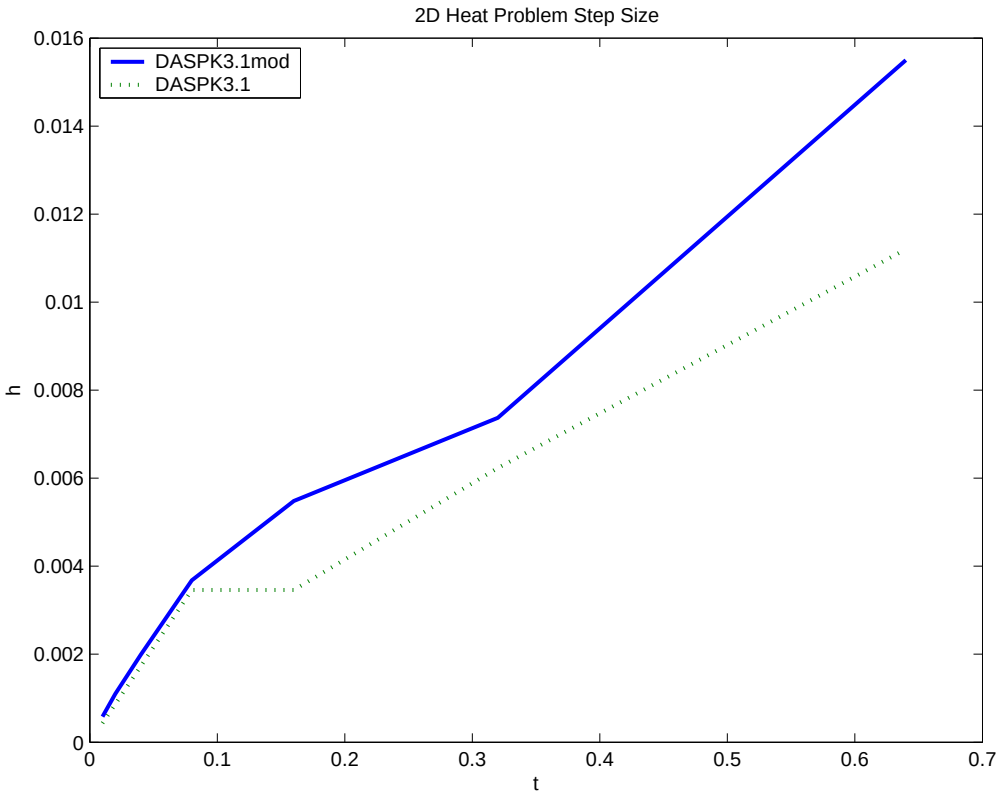


Figure 4.2: 2D Heat Problem Stepsize vs. Time

The second problem is a food web model that describes the interaction and diffusion of a predator species and a prey species on a two-dimensional unit square[6]. The boundary conditions are Neumann, with the normal derivatives set equal to zero. The system of partial differential equations for the concentration vector $c = (c^1, c^2)^T$, in which c^1 is the concentration of prey and c^2 is the concentration of predators, is given by

$$\begin{aligned} \frac{\partial c^1}{\partial t} &= f_1(x, y, t, c) + d_1(c^1_{xx} + c^1_{yy}) \\ 0 &= f_2(x, y, t, c) + d_2(c^2_{xx} + c^2_{yy}) \end{aligned} \tag{4.2}$$

where

$$f_i(x, y, t, c) = c^i * (b_i + \sum_{j=1}^{2} a_{ij} * c^j), \quad i = 1, 2. \tag{4.3}$$

The interaction and diffusion coefficients (a_{ij}, b_i, d_i) were chosen to be

$$A = (a_{ij}) = \begin{pmatrix} -1 & -0.5 \cdot 10^{-6} \\ 10^4 & -1 \end{pmatrix}, \tag{4.4}$$

$$b_1 = -b_2 = 1 + \alpha xy + \beta sin(4\pi x) sin(4\pi y), \tag{4.5}$$

$$\text{and } d_1 = 1,\ d_2 = 0.05 \tag{4.6}$$

where $\alpha = 50$ and $\beta = 100$ are rate parameters. The solutions and sensitivity derivatives, $\frac{\partial c}{\partial \alpha}$ and $\frac{\partial c}{\partial \beta}$, shown in Table 4.2 and Figures 4.3-4.4 were computed from initial conditions

$$\begin{aligned} c^1 &= i * 10 + 16 * x^2 * (1 - x)^2 * 16 * y^2 * (1 - y)^2 \\ c^2 &= 1e5. \end{aligned} \tag{4.7}$$

on a 20 x 20 grid over the time interval $[0, 5]$. The solutions differed by at most 6.7e−5 and the sensitivities differed by at most 1.4.

	DASPK3.1mod	**DASPK3.1**
$\sum_i c^i$	$0.844169374E{+}09$	$1.28468248E+09$
$\sum_i \frac{\partial c}{\partial dprey}^i$	$1.68833875E+09$	$2.56936495E+09$
$\sum_i \frac{\partial c}{\partial dpred}^i$	$2.53250812E+09$	$3.85404743E+09$

Table 4.2: Food Web Population Problem Solution

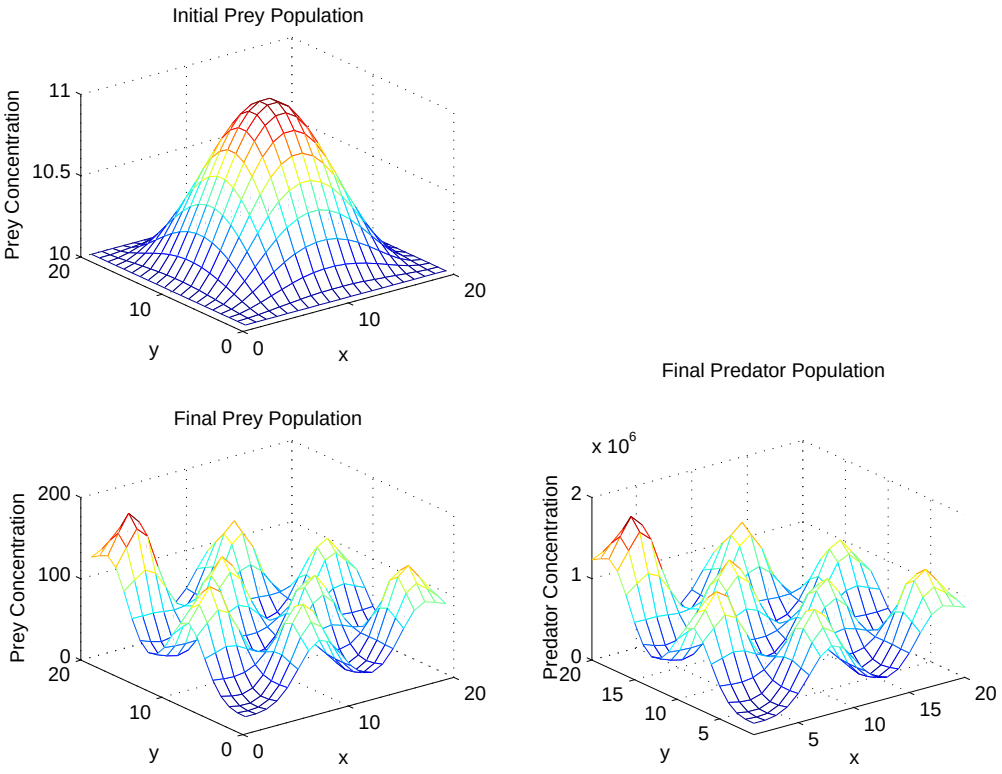


Figure 4.3: Food Web Solution

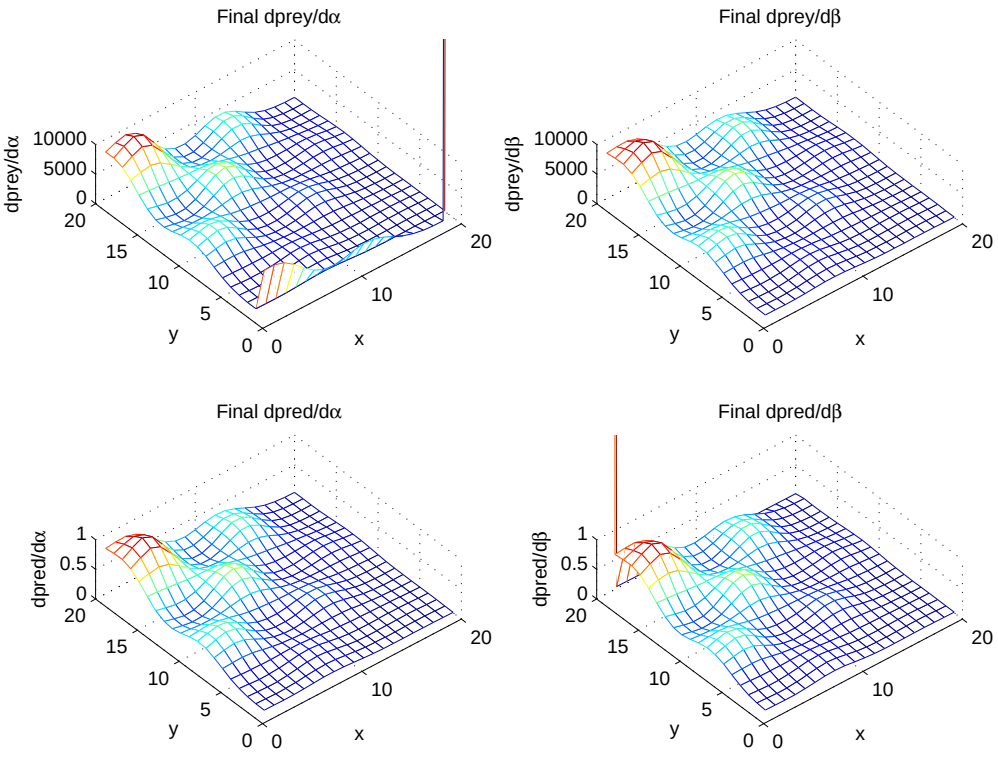


Figure 4.4: Food Web Sensitivities

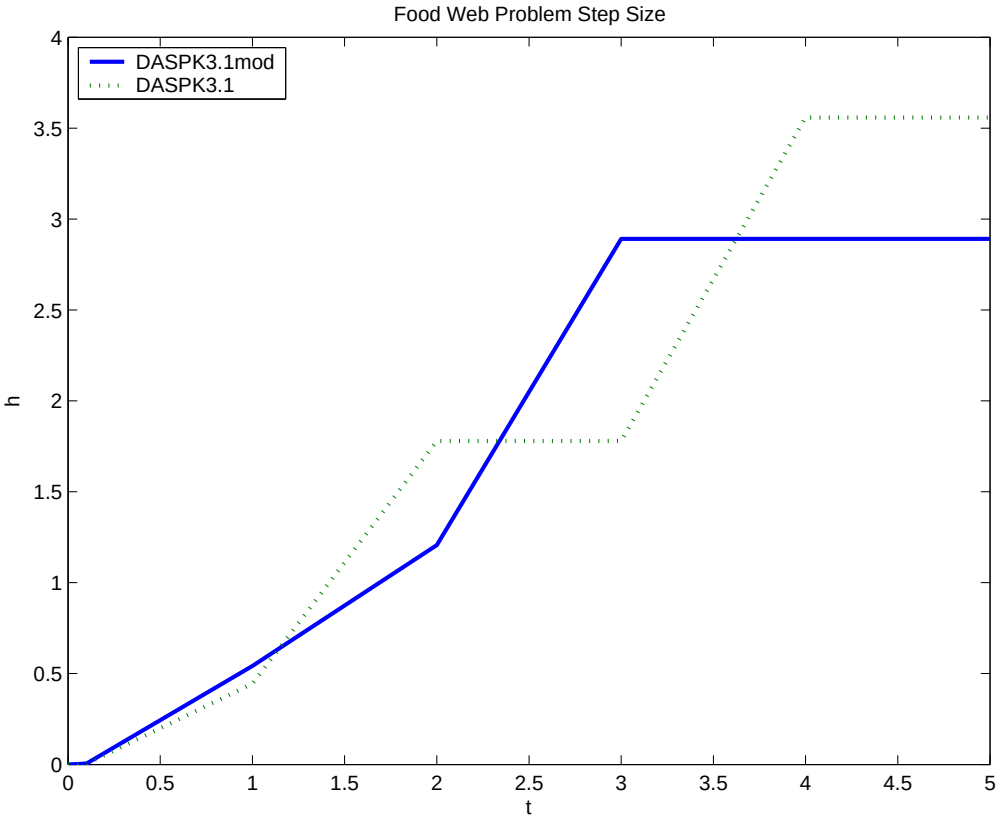


Figure 4.5: Food Web Problem Stepsize vs. Time

Chapter 5

Optimization Results

This chapter contains the results, using both DASPK3.1 and DASPK3.1mod, of three optimal control computation problems: the two-dimensional heat equation, the insertion of a spacecraft into a halo orbit, and the biochemical heat-shock response of *Escherichia coli* bacteria. The first two optimal control problems were solved using COOPT[23]. The last, the heat-shock response problem, was solved using KNITRO[26] because COOPT failed to converge. This may be because the cost functional is not smooth in the heatshock region and KNITRO is able to handle it better due to its trust region approach. Both optimization solvers used DASPK3.1 and DASPK3.1mod to solve the DAE system with constraints.

Table 5.1 shows the performance of the two solvers on these problems. The major iterations, reported by SNOPT[12], generate the search sequence of parameters that converges to the optimal control solution. The accuracy of the sensitivities computed by DASPK affects the convergence of this search sequence. The next column in the table reports the number of iterations taken to find the optimal solution for the parameter sets chosen in the major iterations. Fewer major iterations result in fewer iterations to find the optimal solution over all parameter sets. The heat-shock problem demonstrated significant reduction in the number of iterations and in the corresponding run time. The other two problems: two-dimensional heat equation and halo orbit insertion, had effectively no change in their performance. This may have been because they took very few major it-

erations to solve, and the difference in performance due to smoother sensitivities was masked by the overhead involved in initializing the computation. Also, it may be that these problems were not as sensitive to noise in the sensitivity derivatives. The convergence of the heat-shock problem becomes very slow before it completes, leading us to conjecture that it is sensitive to noise in the sensitivity derivatives.

Problem	**Major Iterations**	**Iterations**	**Time (s)**	**Optimality Tolerance**	**Objective**
heat2d	7/7	931/938	11.31/13.97	5e-2	0.2626373256/0.2626204324
haloIN	5/5	39/39	3.02/3.0	1e-2	1.2417898418/1.2417898418
heat shock stage 1		1000/1000	1729/1623		10.477848747/10.479759936
heat shock stage 2		200/4000**	116/3622**		10.477822686/10.478436909

*DASPK3.1mod/DASPK3.1 **Iteration limit reached.

Table 5.1: Comparison of Solver Performance on Optimization Problems

The 2-D heat optimal control problem is similar to the sensitivity problem in the previous chapter, but has an additional inhomogeneous term, an anisotropic heat flow from the left and bottom boundaries.

$$\frac{dw}{dt} = \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{1}{2} exp^{-\frac{\beta_1}{\beta_2 + w}}, \tag{5.1}$$

where $\beta_1 = 0.2$ and $\beta_2 = 0.05$. The initial condition was $w(x, y, t = 0) = 0$. A uniform 8 x 8 mesh was used to discretize the partial differential equation (PDE) (5.1) with a standard central finite difference approximation. The control parameter u specifies the heat at the bottom and left boundaries. The top and right boundaries were implemented with zero Dirichlet boundary conditions. The tolerances were taken to be $rtol = atol = 1e-6$ for the state equations, $1e-4$ for feasibility, and $5e-2$ for optimality.

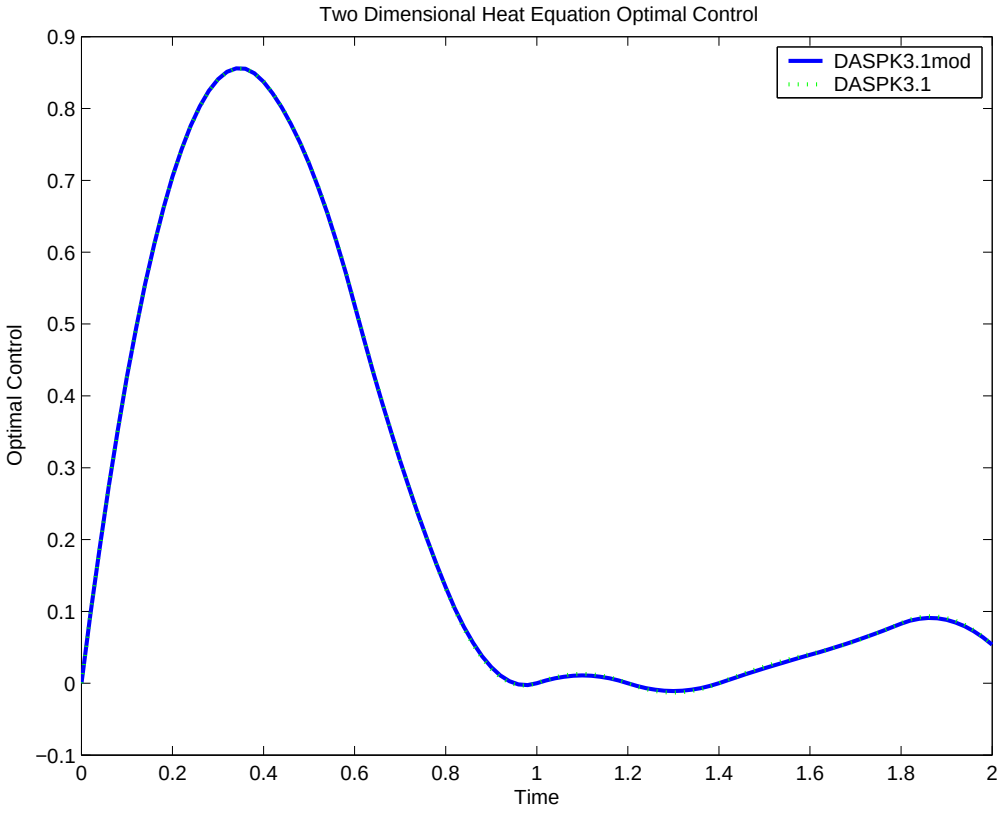


Figure 5.1: Two Dimensional Heat Equation Optimal Control

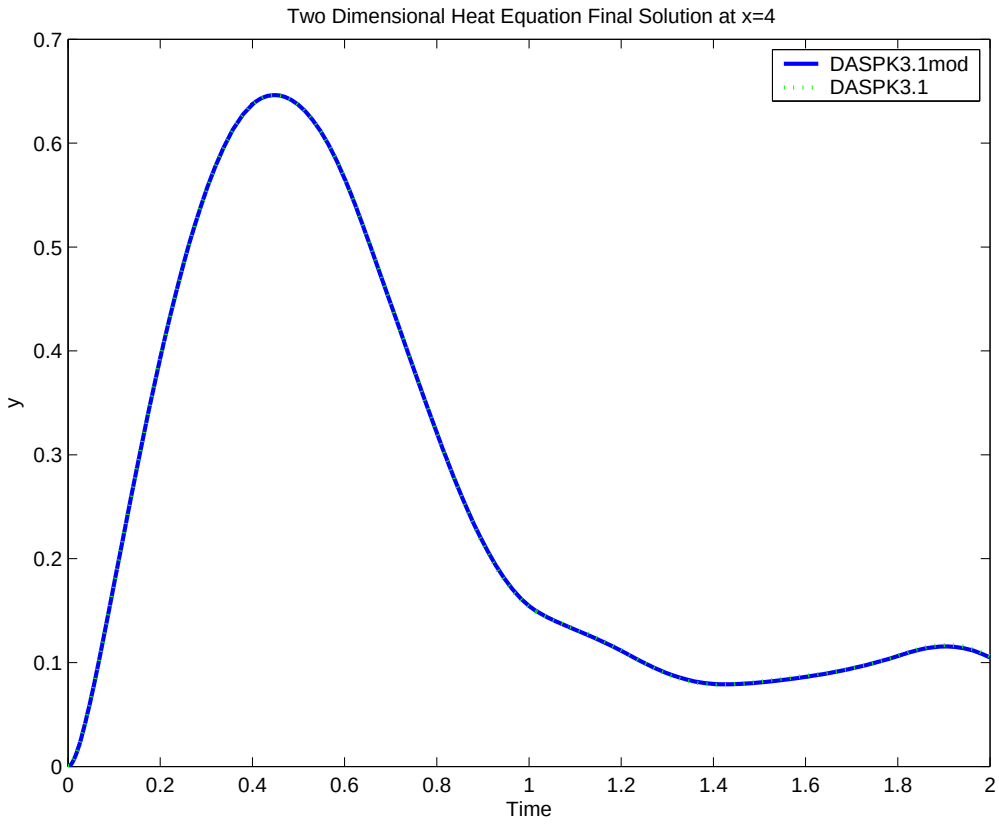


Figure 5.2: Two Dimensional Heat Equation Solution

The halo orbit insertion problem[22] seeks the optimal trajectory correction maneuvers $M_1, M_2, \ldots, M_n$ to insert a spacecraft into a Halo orbit. The optimal cost was computed for combinations of maneuver times $T_i, i = 1, 2, \ldots, n$ and perturbations in the launch velocity. The equations of motion used were from the circular restricted three-body model. This model assumes that the Earth and Sun move in circular orbits about the center of mass of the system, and that the spacecraft has no influence on their orbits. The equations of motion are given by

$$\dot{x}_1 = x_4 \tag{5.2}$$

$$\dot{x}_2 = x_5 \tag{5.3}$$

$$\dot{x}_3 = x_6 \tag{5.4}$$

$$\dot{x}_4 = 2x_2 + \frac{\partial U}{\partial x_1} \tag{5.5}$$

$$\dot{x}_4 = -2x_1 + \frac{\partial U}{\partial x_2} \tag{5.6}$$

$$\dot{x}_4 = \frac{\partial U}{\partial x_3} \tag{5.7}$$

$$U = \frac{1}{2}(x_1^2 + x_2^2) + \frac{1-\mu}{d_1} + \frac{\mu}{d_2} \tag{5.8}$$

$$d_1 = ((x_1 + \mu)^2 + x_2^2)^{1/2} \tag{5.9}$$

$$d_1 = ((x_1 - 1 + \mu)^2) + x_2^2 + x_3^2)^{1/2}. \tag{5.10}$$

The state vector $\mathbf{x} = [x, y, z, v_x, v_y, v_z]^T$, and $\mu = 3.03591 \cdot 10^{-6}$ is the ratio of the Earth's mass to the mass of the Sun-Earth system. There were two position constraints: continuity between maneuvers, and the final position should lie on the halo orbit:

$$x_i^p(T_i) = x_{i+1}^p(T_i), \quad i = 1, 2, \ldots, n-1 \tag{5.11}$$

$$x_n^p(T_n) = x_H^p(T_n). \tag{5.12}$$

Time constraints were that the first maneuver be delayed by at least $TCM1_{min}$, and that the order of maneuvers is preserved

$$T_1 \geq TCM1_{min} \tag{5.13}$$

$$T_{i-1} < T_i < T_{i+1}, \quad i = 1, 2, \ldots, n-1. \tag{5.14}$$

Finally, the cost function was defined as the sum of the velocity discontinuities due to the maneuvers

$$\Delta \mathbf{v}_i = \mathbf{x}^v_{i+1}(T_i) - \mathbf{x}^v_i(T_i), \quad i = 1, 2, \ldots, n-1, \tag{5.15}$$

$$\Delta \mathbf{v}_n = \mathbf{x}^v_H(T_n) - \mathbf{x}^v_n(T_n). \tag{5.16}$$

The results presented here are for a 3-maneuver flight with initial conditions $\mathbf{x} = [1.000035565608365e+00 \text{ AU}, 1.298950527135473e-05,$ $-1.657172577465346e-05, -1.547585875645079e-01 \text{ AU/rad},$ $3.157800035860918e{-}01, -1.167438053370118e{-}01]^T$, a launch velocity perturbation of 1.007236004475729e-04 AU/rad, and $TCM1_{min} = 0.2$ rad. The tolerances are $rtol = 1e-9$ and $atol = 1e-10$ for the state equations, $rtol = 1e-8$ and $atol = 1e-9$ for the sensitivities, feasibility is $1e-6$, and optimality is $1e-2$.

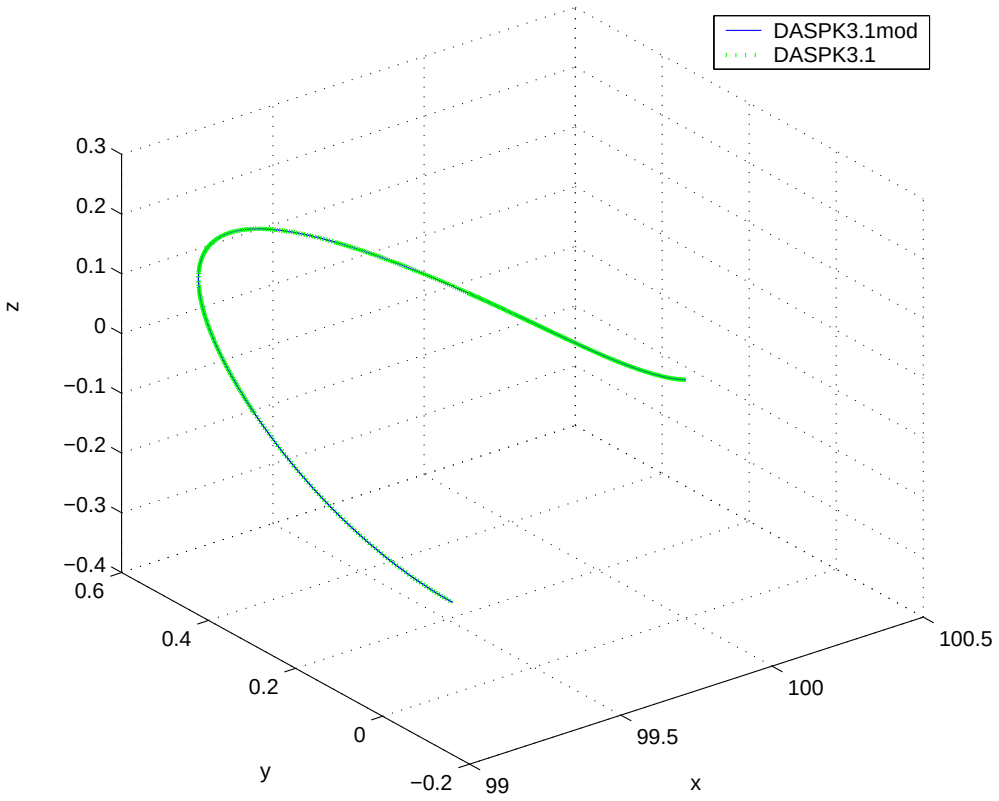


Figure 5.3: Halo Orbit Insertion Solution

The heat shock problem models the production of heat shock proteins in *Escherichia coli* cells as a response to extreme heat. These special proteins refold other cell proteins, essential to cell survival, that have become unfolded due to excessive heat. The optimization problem is to find a balance between the effect of the protective proteins and the cost of their production. The mathematical model of the heat shock response is reported in [9]. The model employs first order kinetics (law of mass-action) to describe the various components of the heat shock system, namely the synthesis of heat shock proteins and their regulator σ_{32}, in addition to protein folding and the association/dissociation activity of molecules. Reactions involving association/dissociation of molecules are often faster than synthesis and degradation of new molecules. Therefore the model exhibits a wide range of time scales, hence it is subject to numerical stiffness. The differential equations that describe the fast states were transformed into algebraic constraints, through a partial equilibrium approximation, yielding an index-one system of differential algebraic equations (DAE). The DAE system describes the evolution of 11 differential variables, coupled to the evolution of 20 algebraic variables, and includes 32 parameters. [16] The objective function was given by

$$\int_0^T x_1^2(t,\theta) + \alpha x_2^2(t,\theta)dt, \tag{5.17}$$

where x_1 and x_2 are the unfolded and heat shock proteins respectively, θ is a vector of the model parameters, and α is the relative weight of the two terms. The optimization problem was solved in two stages. The first stage was run for a fixed number of iterations, to reach a steady-state solution. The second stage was initialized with both this steady-state solution and the optimal parameters found in the first stage. This problem was solved using KNITRO[26]. The tolerances are $rtol = atol = 1e-6$ for the state equations and for the sensitivities, relative feasibility tolerance is $1e-6$, and relative optimality tolerance is $1e-6$. The KNITRO initial trust region radius is $1e-2$, the initial pivot threshold is $5e-1$, and the initial barrier parameter value is $1e-1$.

Fig. 5.4 shows the Pareto optimal curve (red) generated by the solution of the optimization problem. The plot also shows the cost of chaperons and un-

folded proteins for wild type heat shock in *E. coli*. The closeness of this point to the optimal curve indicates that the heat shock response is operating in a near-optimal regime, perhaps as a result of a long evolutionary past that converged to a near-optimal solution. To check that our finding is substantial and rule out the possibility that the structure of the model itself is generating near-optimal solutions for all parameter values, we randomly generated combinations of the parameters and computed their costs. These randomly chosen parameter combinations mostly yielded operating points well inside the non-optimal region, therefore substantiating our conclusions. Three illustrative points are shown in Figure 5.4. [16]

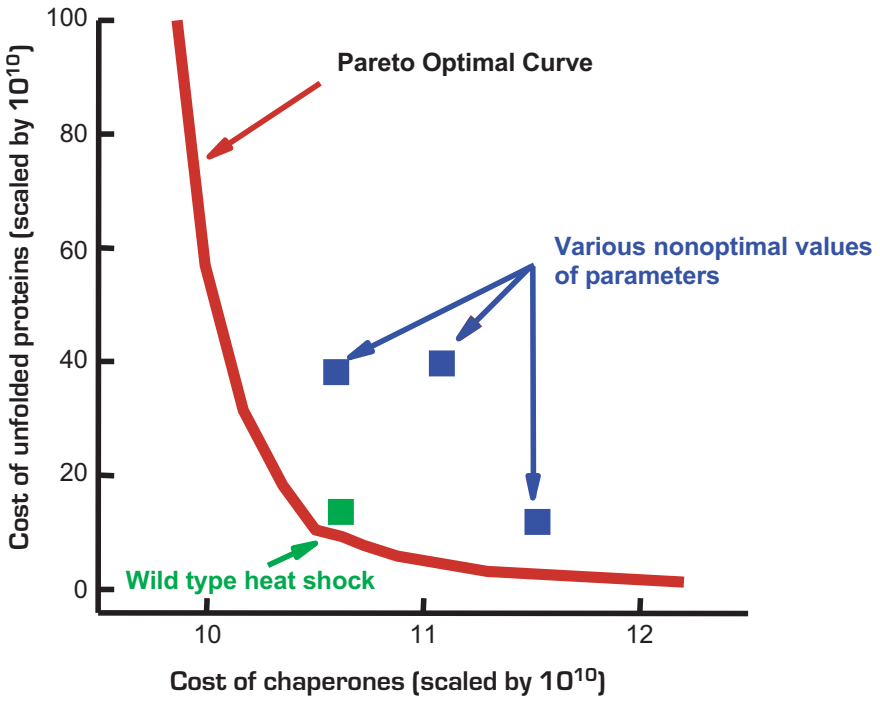


Figure 5.4: Heat Shock Optimization Results

The performance in the first stage, computing the transient response, is shown in Figure 5.5. In this stage, the optimizer was run for one thousand optimization

iterations to reach a steady-state solution. For every value of the weighting parameter α, DASPKmod took fewer steps. However, that did not translate into shorter execution times for the $\alpha = 100$ and $\alpha = 2000$ cases. In these cases more time may have been spent at each time step reaching a solution in the Newton iteration.

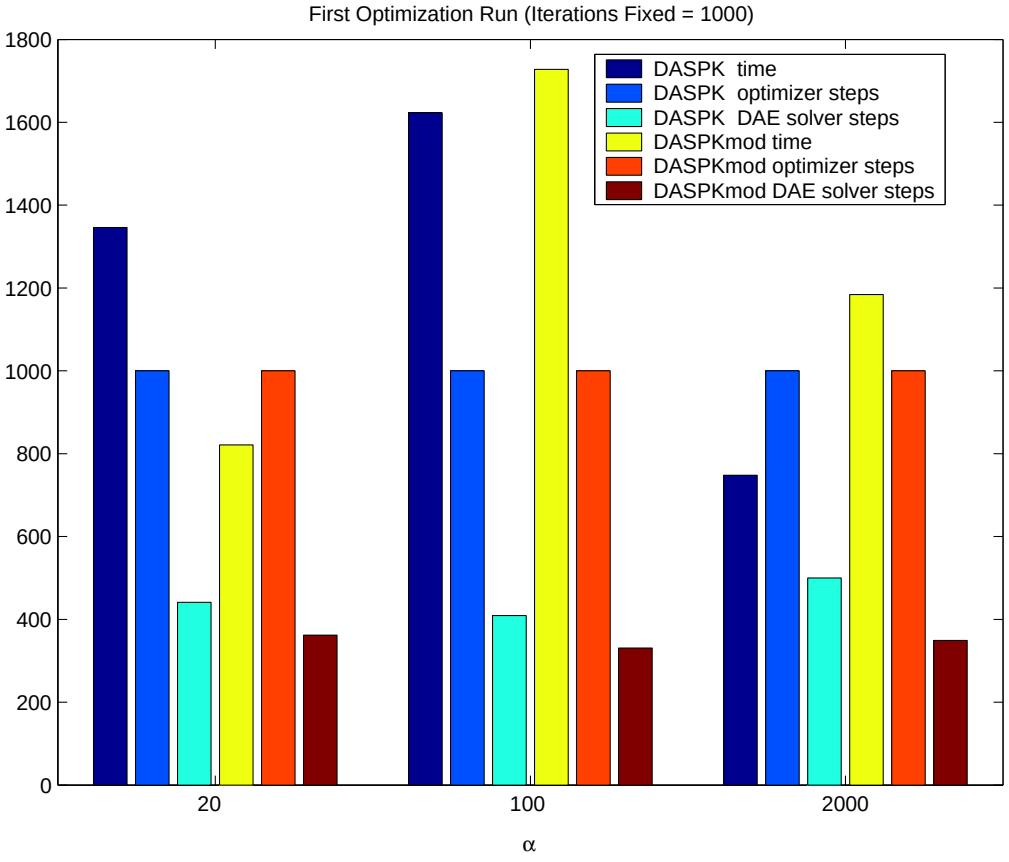


Figure 5.5: Heat Shock Solver Performance for First Stage

The performance in the second stage, computing the optimal solution, is shown in Figure 5.6. In all cases, the number of optimization steps when using DASPKmod was significantly lower than when using DASPK. This was not true for the number of DASPK steps. However, the execution time appears to be influenced primarily by the number of optimization steps, not the number of DASPK steps.

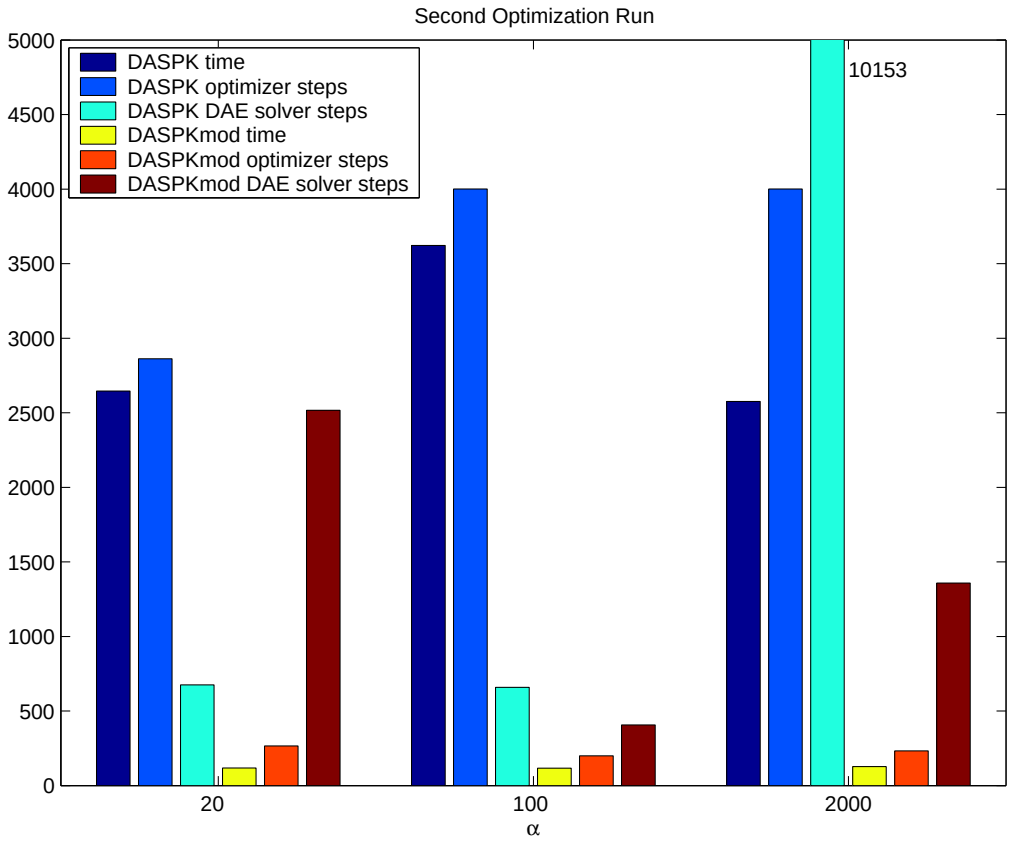


Figure 5.6: Heat Shock Solver Performance for Second Stage

Though there was no appreciable performance improvement when solving the heat equation and orbit insertion problems, the significant reduction in the number of optimizer steps and the execution time when solving the heat shock problem are promising results for the new stepsize control in DAE solution, within the context of dynamic optimization.

Chapter 6

Conclusions

The reduction in execution time when solving the heat shock problem, from an hour to a few minutes, is a dramatic improvement. We conjecture that the performance improvement is due to the smoother behavior of the solution components with respect to perturbation of the problem parameters. Though a performance improvement was not observed for all the problems attempted, neither was there a degradation in performance. There may be problems that fail to converge at all using the simpler linear stepsize controller, and that could be solved using this improved stepsize controller. This approach is widely applicable to derivative-based optimizers, and holds promise for solving difficult optimization problems that are sensitive to noise in functions computed by an ODE or DAE solver.

Bibliography

[1] `http://pitagora.dm.uniba.it/~testset/`.

[2] U. Ascher and L. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.

[3] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR-Generating derivative codes from Fortran programs. *Scientific Programming*, 1992.

[4] BR/ARLO-CRC. *CBS-reaction-meeting Köln. Handouts*, May 1993.

[5] K. Brenan, S. Campbell, and L. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, second edition, 1996.

[6] P. N. Brown. Decay to uniform states in food webs. *SIAM J. Appl. Math.*, 46:376–392, 1986.

[7] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM J. Sci. Comput.*, 15:1467–1488, 1994.

[8] R. Byrd, M. E. Hribar, and J. Nocedal. An Interior Point Method for Large Scale Nonlinear Programming. *SIAM J. Optimization*, 9(4), 1999.

[9] H. El-Samad, H. Kurata, J.C. Doyle, C.A. Gross, and M. Khammash. Surviving heat shock: control strategies for robustness and performance. In *Submitted, PNAS*, 2004.

[10] W. Feehery, J. Tolsma, and P. Barton. Efficient sensitivity analysis of large-scale differential-algebraic systems. *Applied Numerical Mathematics*, 25:41–54, 1997.

[11] P. Gill, L. Jay, M. Leonard, L. R. Petzold, and V. Sharma. An SQP Method for the Optimal Control of Large-Scale Dynamical Systems. *J. Comp. Appl. Math.*, 120:197–213, 2000.

[12] P. Gill, W. Murray, and M. Saunders. User's Guide for SNOPT Version 6: A Fortran Package for Large-Scale Nonlinear Programming. Technical report, Systems Optimization Lab, Stanford University, 2002.

[13] S. Li and L. Petzold. Design of New DASPK for Sensitivity Analysis. Technical report, University of California, Santa Barbara, 1999.

[14] S. Li and L. Petzold. Software and Algorithms for Sensitivity Analysis of Large-Scale Differential Algebraic Systems. Technical report, University of California, Santa Barbara, 2000.

[15] F. Mazzia and F. Iavernaro. Test Set for Initial Value Problem Solvers. Technical Report 40/2003, Bari University, Bari, Italy, 2003.

[16] K. Meeker, C. Homescu, L. Petzold, H. El-Samad, and M. Khammash. Digital Filter Stepsize Control in DASPK and its Effect on Control Optimization Performance. In *Submitted*, 2004.

[17] L. R. Petzold. *A description of DASSL: A differential/algebraic system solver.* R. S. Stepleman et al. (Eds.), North-Holland, Amsterdam, 1983.

[18] E. Schäfer. A new approach to explain the 'high irradiance responses' of photomorphogenesis on the basis of phytochrome. *J. of Math. Biology*, 2:41–56, 1975.

[19] G. Söderlind. Digital filters in adaptive time-stepping. *ACM T Math Software*, 29:1–26, 2003.

[20] G. Söderlind and L. Wang. Evaluating Numerical ODE/DAE Methods, Algorithms and Software. *J Comp Methods in Sci Eng*, 2:1–3, 2001.

[21] G. Söderlind and L. Wang. Adaptive time-stepping and computational stability. *ACM T Comp Logic*, V:1–14, 2002.

[22] R. Serban, W.S. Koon, M. Lo, J. Marsden, L.R. Petzold, S.D. Ross, and R. Wilson. Halo Orbit Mission Correction Maneuvers Using Optimal Control. *Automatica*, 38(4):571–583, 2002.

[23] R. Serban and L. R. Petzold. COOPT - a software package for optimal control of large-scale differential-algebraic equation systems. *Math Comput Simulat*, 56:187–203, 2001.

[24] R. van der Hout. Private communication. from Akzo Nobel Central Research, 1994.

[25] J.G. Verwer. Gauss-Seidel iteration for stiff ODEs from chemical kinetics. *SIAM J. Sci. Comput.*, 15(5):1243–1259, 1994.

[26] R. A. Waltz and J. Nocedal. KNITRO User's Manual Technical Report OTC 2003/05. Technical report, Optimization Technology Center, Northwestern University, Evanston, IL, 2003.

Appendix A

Code Changes

A.1 DASPK3.1 Original Stepsize Controller

```
C-----------------------------------------------------------------------
C     BLOCK 5
C     The step is successful. Determine
C     the best order and stepsize for
C     the next step. Update the differences
C     for the next step.
C-----------------------------------------------------------------------

C
C     If IPHASE = 0, increase order by one and multiply stepsize by
C     factor two
C
545   K = KP1
      HNEW = H*2.0D0
      H = HNEW
      GO TO 575
C
C
C     Determine the appropriate stepsize for
C     the next step.
C
550   HNEW=H
```

```
      TEMP2=K+1
      R=(2.0D0*EST+0.0001D0)**(-1.0D0/TEMP2)
      IF(R .LT. 2.0D0) GO TO 555
      HNEW = 2.0D0*H
      GO TO 560
555   IF(R .GT. 1.0D0) GO TO 560
      R = MAX(0.5D0,MIN(0.9D0,R))
      HNEW = H*R
560   H=HNEW
C
C
C     Update differences for next step
C
575   CONTINUE



C-----------------------------------------------------------------------
C     BLOCK 6
C     The step is unsuccessful. Restore X,PSI,PHI
C     Determine appropriate stepsize for
C     continuing the integration, or exit with
C     an error flag if there have been many
C     failures.
C-----------------------------------------------------------------------

C
C     On first error test failure, keep current order or lower
C     order by one.  Compute new stepsize based on differences
C     of the solution.
C
      K = KNEW
      TEMP2 = K + 1
      R = 0.90D0*(2.0D0*EST+0.0001D0)**(-1.0D0/TEMP2)
      R = MAX(0.25D0,MIN(0.9D0,R))
      H = H*R
      IF (ABS(H) .GE. HMIN) GO TO 690
```

```
      IDID = -6
      GO TO 675
```

A.2 DASPK3.1mod Digital Filter Stepsize Controller

```
CWL
CWL  GET_H --  computes the new stepsize. There are three controllers
CWL            one can choose: H211b(b=4), PI.4.2 and standard
CWL            control.
CWL  CTRNM -- the name of controller:
CWL            CTRNM = H211B4:   H221B(B=4) controller
CWL                  = PI42:     PI.4.2 controller
CWL                  = STAND:    standard controller
CWL                  default controller is H211b4
CWL                  if one wants to use the other controller
CWL                  One has to change this parameter.
CWL  DLIMITER -- function of nonlinear smooth limiter.
CWL  DKAPPA -- parameter for nonliner smooth limiter. usually chosen
CWL            0.7 < dkappa < 2.0
CWl            defaultvalue is dkappa = 1.0
CWL




CWL
      CHARACTER*10  CTRNM
CWL
C
C
C***FIRST EXECUTABLE STATEMENT  DDASPK
C
CWL   give the name of stepsize controller and parameter of
CWL   nonlinear smooth limiter
      CTRNM = 'H211B4'
      DKAPPA = 1.0D0
```

```
CWL
CWL



C-----------------------------------------------------------------------
C     BLOCK 5
C     The step is successful. Determine
C     the best order and stepsize for
C     the next step. Update the differences
C     for the next step.
C-----------------------------------------------------------------------



C
C     If IPHASE = 0, increase order by one and multiply stepsize by
C     factor two
C
545   K = KP1
      IF(JSTART .EQ. 0) ESTOLD = EST
550   TEMP2=K+1
      CALL GET_H(EST,ESTOLD,RATIO,H,HNEW,TEMP2,CTRNM,DKAPPA)
560   H=HNEW
C
C
C     Update differences for next step
C
575   CONTINUE



C-----------------------------------------------------------------------
C     BLOCK 6
C     The step is unsuccessful. Restore X,PSI,PHI
C     Determine appropriate stepsize for
C     continuing the integration, or exit with
```

```
C     an error flag if there have been many
C     failures.
C-----------------------------------------------------------------------




C
C     On first error test failure, keep current order or lower
C     order by one.  Compute new stepsize based on differences
C     of the solution.
C
      K = KNEW
      TEMP2 = K + 1
      CALL GET_H(EST,ESTOLD,RATIO,H,HNEW,TEMP2,CTRNM,DKAPPA)
      H = HNEW
      IF (ABS(H) .GE. HMIN) GO TO 690
      IDID = -6
      GO TO 675



      SUBROUTINE  GET_H(EST,ESTOLD,RATIO,H,HNEW,TK,CTRNM,DKAPPA)
      IMPLICIT DOUBLE PRECISION(A-H, O-Z)
      CHARACTER*(*) CTRNM
C
C
      IF (CTRNM .EQ. 'H211B4') THEN
         B1 = 0.25D0/TK
         B2 = 0.25D0/TK
         A2 = 0.25D0
      ELSEIF (CTRNM .EQ. 'PI42')  THEN
         B1 = 0.6D0/TK
         B2 = -0.2D0/TK
         A2 = 0.0D0
      ELSE
         B1 =  1.0D0/TK
         B2 =  0.0D0
```

```
         A2 =  0.0D0
      ENDIF
C
C      WRITE(*,*) "DURING",EST,ESTOLD,RATIO,RHO
      RATIO =(2.0D0*EST+1.0D-8)**(-B1)*
     *     (2.0D0*ESTOLD+1.0D-8)**(-B2)*
     *     RATIO**(-A2)
      RHO = DLIMITER(RATIO,DKAPPA)
C      WRITE(*,*) "DURING",EST,ESTOLD,RATIO,RHO
      HNEW = H*RHO
C
      RETURN
      END
```

A.3 DASPK3.1mod Newton Termination Criteria and Freeing Order

```
C-----------------------------------------------------------------------
C     This block is executed on the initial call only.
C     Set the initial step size, the error weight vector, and PHI.
C     Compute unknown initial components of Y and YPRIME, if requested.
C-----------------------------------------------------------------------

CWL
CWL   Here we change the Newton termination criterion
CWL   from 0.33 to 0.033
C      RWORK(LEPCON) = 0.33D0
      RWORK(LEPCON) = 0.033D0
CWL



CWL   We will free the order and error sequence decide whether or
CWL   not the order will be increased
C      IF(KP1.GE.NS.OR.KDIFF.EQ.1)GO TO 550
CWL
```