T U M

FAKULTÄT FÜR MATHEMATIK

Documentation of PAREST — A Multiple Shooting Code For Optimization Problems in Differential-Algebraic Equations

Alexander Heim
Oskar von Stryk

TUM–M9616
November 1996

TECHNISCHE UNIVERSITÄT MÜNCHEN

TUM-MATH-11-1996-M9616-200/1.-FMI
Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©1996 FAKULTÄT FÜR MATHEMATIK UND
INSTITUT FÜR INFORMATIK
TECHNISCHE UNIVERSITÄT MÜNCHEN

Typescript: ---

Druck: Fakultät für Mathematik und
Institut für Informatik der
Technischen Universität München

Documentation of ParEst — A Multiple Shooting Code For Optimization Problems in Differential-Algebraic Equations

Alexander Heim[1]
and
Oskar von Stryk[2]
Lehrstuhl für Höhere Mathematik und Numerische Mathematik
Technische Universität München
D-80290 München

Contents

1 Purpose, Problem Descriptions, and Notation **3**

1.1 Abstract 3

1.2 Dynamic model 4

1.3 Parameter estimation problems 5

1.4 Optimal control problems 6

1.5 Multiple shooting approach 6

1.6 Survey on the three time grids 7

2 Fortran Subroutines of the Model Functions **8**

2.1 `DAGFY` for the computation of f^y (differential equations) 8

2.2 `DAGFV` for the computation of f^v (algebraic equations) 9

2.3 `DAGRB` for the computation of the boundary conditions r . . . 9

2.4 `MESSH` for the computation of the measurement functions h . . 10

[1] heim@mathematik.tu-muenchen.de, URL: http:\\www.mathematik.tu-muenchen.de\~heim\
[2] stryk@mathematik.tu-muenchen.de, URL: http:\\www-home.mathematik.tu-muenchen.de\~stryk\

2.5 `ZSTDB` for the computation of the (nonlinear) inequality constraints S 10

2.6 `ZLFKT` for the computation of the optimal control objective Φ . 11

3 Input File of ParEst **12**

3.1 Section GENERAL 13

3.2 Section NAMEN 16

3.3 Section MESSWERTE 16

3.4 Section GRENZEN 17

3.5 Section SKALIERUNG 18

3.6 Section STARTWERTE 19

3.7 Section STEUERUNG 21

4 The Programs `parest` and `pgraph` **22**

5 Examples **27**

5.1 Parameter identification of a planar pendulum 27

5.2 A minimum energy problem 28

References **30**

1 Purpose, Problem Descriptions, and Notation

1.1 Abstract

ParEst is a direct multiple shooting method designed to solve parameter estimation and optimal control problems in systems of differential-algebraic equations or ordinary differential equations [1], [2].

In parameter estimation problems (also called parameter identification problems), parameters of the dynamic model are fitted to a set of measurements of (functions of) the state variables by minimizing a nonlinear least squares objective.
In optimal control problems the control variables are computed which minimize a performance index.
In both cases, inequality constraints may be imposed on the state and control variables as well.

ParEst is a direct transcription method. I. e. a multiple shooting approach is used to discretize and solve the boundary value problem. The control variables are discretized by piecewise linear functions.
Generalized Gauss-Newton methods [12] and Sequential Quadratic Programming methods [9] are used to solve the resulting nonlinearly constrained least squares problems and nonlinearly constrained optimization problems.

Version 1.0 of the software consists of a numerical core written in Fortran 77 and a main program written in C for handling input and output. The user must provide subroutines that define the model functions (objective, differential equations, boundary conditions, nonlinear inequality constraints). This is usually done by editing the pre-defined subroutines in the file **uspec.f** (Fortran 77). Further informations on the problem (as the dimensions of the problem, specifications of the numerical methods, lower and upper bounds for all variables, optional scaling, the three time grids (of the multiple shooting discretization, of the control discretization, and of the times of measurements (only in parameter identification)) as well as initial estimates of state and control variables, the measurement values (only in parameter estimation)) have to be supplied by editing the input file **start.dat**.

A supplementary program PGraph is provided which supports a visualization of the numerical results using the LRZ-graphics library [3].

1.2 Dynamic model

The dynamic model is described by *a system of differential-algebraic equations of index 1 in semi-explicit form*:

$$\begin{array}{rcll} \dot{y} & = & f^y(t,y,v,u,p)\,, & f^y = (f_1^y,\ldots,f_{NY}^y) \\ 0 & = & f^v(t,y,v,u,p)\,, & f^v = (f_1^v,\ldots,f_{NV}^v) \end{array} \tag{1}$$

Notation

differential state variable:	$y(t)$	$=$	$(y_1(t),\ldots,y_{NY}(t))$
algebraic state variable:	$v(t)$	$=$	$(v_1(t),\ldots,v_{NV}(t))$
control variable:	$u(t)$	$=$	$(u_1(t),\ldots,u_{NU}(t))$
control (or design) parameters:	p	$=$	$(p_1,\ldots,p_{NP})$

(In the case of a dynamical system of only ordinary differential equations NV is zero.)

Further problem dependent functions are:

- *boundary conditions* (including initial and final conditions of the state variables) defined by

$$r(t_0,y(t_0),v(t_0),u(t_0),t_f,y(t_f),v(t_f),u(t_f),p) \;=\; 0\,, \tag{2}$$

  where $r = (r_1,\ldots,r_{NRB})$,

- *(nonlinear) inequality constraints* on the control and state variables

$$\begin{array}{ccccc} a_1 & \leq & S_1(t,y(t),v(t),u(t),p) & \leq & b_1 \\ & \vdots & & \vdots & \\ a_{NZB} & \leq & S_{NZB}(t,y(t),v(t),u(t),p) & \leq & b_{NZB} \end{array} \tag{3}$$

  (Constraints of this kind usually appear more often in optimal control than in parameter estimation problems.)

- (general) *functions of the state (and control) variables*

$$h(t,y(t),v(t),u(t),p)\,, \qquad h = (h_1,\ldots,h_{NH})\,, \tag{4}$$

  which may be used for two different purposes:

  - In *parameter identification* problems the state variables $y(t)$ and $v(t)$ often cannot be measured directly. But functions h of them have been measured in an experiment. The relation between the measured quantities and the state variables is given by the function $h(t,y,v,u,p)$.

– PGraph also generates graphical output of the functions h_i. Therefore any functions h_i of the state and control variables might be used in addition for the purpose of monitoring properties of the computed trajectories (for example, phase diagrams as the altitude/velocity diagram in optimal control problems from flight mechanics, or violations of algebraic index-2- or index-3-constraints if index reduction has been applied to a problem with a system of differential-algebraic equations of a higher index (as in the pendulum problem described in Section 5.1)).

The components of the functions $h(\cdot)$ do only enter the optimization program ParEst if they are required in a parameter estimation problem. Otherwise they will only be used for plotting by PGraph.

1.3 Parameter estimation problems

The problem in parameter identification is to estimate unknown parameters p_i of the dynamic model. (inverse problem). Measurement values z_{ij} of an experiment are given which have been obtained at the l times t_j^e $(j = 1, \ldots, l)$ where $t_1^e < \ldots < t_l^e$.

Measurement values at times t_j^e are quantities depending on

- the differential state variables $y(t_j^e)$,
- the algebraic state variables $v(t_j^e)$ or
- functions of the state variables $h(t_j^e, y(t_j^e), v(t_j^e), u(t_j^e), p)$.

The efficiency of the numerical computations can be improved if the case of directly measured state variables is treated separately from the case that only functions of them have been measured.

Positive real constants s_{ij} can be used in order to weight the deviations from the z_{ij}-values in the nonlinear least squares objective

$$\Phi = \sum_{j=1}^{l} \left(\sum_{\substack{i \\ z_{ij} \sim y}} \frac{(y(t_j^e) - z_{ij})^2}{s_{ij}^2} + \sum_{\substack{i \\ z_{ij} \sim v}} \frac{(v(t_j^e) - z_{ij})^2}{s_{ij}^2} + \sum_{\substack{i \\ z_{ij} \sim h}} \frac{(h(t_j^e, y(t_j^e), v(t_j^e), u(t_j^e), p) - z_{ij})^2}{s_{ij}^2} \right). \tag{5}$$

The parameters p_i have to minimize Φ subject to the differential-algebraic equations (1), the boundary conditions (2), and the inequality constraints (3).

The resulting nonlinear least squares problem with nonlinear constraints can be solved by either a generalized Gauss-Newton or a Sequential Quadratic

Programming Method.

1.4 Optimal control problems

The problem in optimal control is to determine the control variable $u(t)$ and the control (or design) parameters p in order to minimize an objective of Mayer type

$$J[u] = \Phi(t_f, y(t_f), v(t_f), u(t_f), p) \longrightarrow \min! \tag{6}$$

(with a real-valued function Φ). The final time t_f may be givend or free.

In order to obtain an approximation of the optimal control a discretization of the control variable is applied. Hereby, a time grid $\{t_j^s\}$ consisting of *NSGIT* knots

$$t_0 = t_1^s < t_2^s < \cdots < t_{NSGIT}^s = t_f \tag{7}$$

is introduced. The control is approximated by a continuous, piecewise linear function over this time grid. The values of the control variables at these knots $u(t_j^s)$ have to be determined in the optimization.

Also the inequality constraints (3) are discretized by the algorithm in a similiar way. They are satisfied at the times t_j^s of the control discretization grid

$$a \;\leq\; S(t_j^s, y(t_j^s), v(t_j^s), u(t_j^s), p) \;\leq\; b, \quad i = 1, \ldots, NSGIT\,. \tag{8}$$

The resulting nonlinearly constrained minimization problem is solved by a Sequential Quadratic Programming Method.

1.5 Multiple shooting approach

In order to increase the robustness of the numerical method, multiple shooting can optionally be applied to split up the whole time interval into several smaller ones for numerical integration [7], [15]. Therefore, another time grid $\{t_j^m\}$ consisting of *NMZGIT* multiple shooting nodes

$$t_0 = t_1^m < t_2^m < \cdots < t_{NMZGIT}^m = t_f \tag{9}$$

can be selected. The values of the state variables $y(t_j^m)$ and $v(t_j^m)$ at the multiple shooting nodes then become unknowns that have to be determined by the optimization procedure.

A "relaxation" technique is applied in order to obtain *consistent* initial values for the differential-algebraic intial value problems on the subintervals of the multiple shooting discretization [4].

It should be noted that the time grid of the multiple shooting nodes is *independent* of the time grid for the discretization of the control variables.

1.6 Survey on the three time grids

Three time grids may appear (which do not depend on each other):

The measurement grid of a parameter identification problem consists of the times t_j^e of the measurements.

The control grid of a discretized optimal control problem consists of the times t_j^s where a piecewise linear approximation of the control variable $u(t)$ is given by the values $u(t_j^s)$.

Also, any inequality constraint $a \leq S(t, y(t), v(t), u(t), p) \leq b$ of an optimal control problem will only be satisfied at the times of the control discretization grid.

The multiple shooting grid is an optional grid consisting of the multiple shooting nodes t_j^m. Its purpose is to split up the whole integration interval $[t_0, t_f]$ into a sequence of smaller ones $[t_j^m, t_{j+1}^m]$, $j = 1, 2, \ldots$.

Please note that the **multiple shooting grid** and (if $NU \geq 0$) also the **control discretization grid** (in optimal control problems) always must contain **at least two** grid points, namely initial time t_0 and final time t_f.

2 Fortran Subroutines of the Model Functions

The previously described functions define the specific problem. For the numerical computation of them a set of problem dependent subroutines in Fortran 77 has to be provided by the user. They are collected in the file **uspec.f**.

Important note:
Parameters of the problem dependent subroutines in the file **uspec.f** denoted by "Ausgabe" or "Output" **have to be set** on exit of the subroutine.
All other parameters which are **not explicitly denoted** by "Ausgabe" or "Output" **may not be changed** in a subroutine. Changing them will be harmful!

2.1 `DAGFY` for the computation of f^y (differential equations)

function $f^y(t, y(t), v(t), u(t), p)$ of Eq. (1)

syntax `SUBROUTINE DAGFY (NY,NV,NU,NP,T,Y,V,U,P,FY)`

parameters `IMPLICIT NONE`

`INTEGER NY`	dimension of $y(t)$
`INTEGER NV`	dimension of $v(t)$
`INTEGER NU`	dimension of $u(t)$
`INTEGER NP`	dimension of p
`DOUBLE PRECISION T`	time t
`DOUBLE PRECISION Y(NY)`	value of $y(t)$
`DOUBLE PRECISION V(NV)`	value of $v(t)$
`DOUBLE PRECISION U(NU)`	value of $u(t)$
`DOUBLE PRECISION P(NP)`	value of p
`DOUBLE PRECISION FY(NY)`	resulting value of f^y ($\rightarrow$ Ausgabe/Output)

2.2 `DAGFV` for the computation of f^v (algebraic equations)

function $f^v(t, y(t), v(t), u(t), p)$ of Eq. (1)

syntax `SUBROUTINE DAGFV (NY,NV,NU,NP,T,Y,V,U,P,FV)`

parameters `IMPLICIT NONE`

`INTEGER NY`	dimension of $y(t)$
`INTEGER NV`	dimension of $v(t)$
`INTEGER NU`	dimension of $u(t)$
`INTEGER NP`	dimension of p
`DOUBLE PRECISION T`	time t
`DOUBLE PRECISION Y(NY)`	value of $y(t)$
`DOUBLE PRECISION V(NV)`	value of $v(t)$
`DOUBLE PRECISION U(NU)`	value of $u(t)$
`DOUBLE PRECISION P(NP)`	value of p
`DOUBLE PRECISION FV(NY)`	resulting value f^v ($\rightarrow$ Ausgabe/Output)

2.3 `DAGRB` for the computation of the boundary conditions r

function $r(t_0, y(t_0), v(t_0), u(t_0), t_f, y(t_f), u(t_f), s(t_f), p)$ of Eq. (2)

syntax `SUBROUTINE DAGRB (NY,NV,NU,NP,NRB,T0,Y0,V0,U0,`
`TF,YF,VF,UF,P,RB)`

parameters `IMPLICIT NONE`

`INTEGER NY`	dimension of $y(t)$
`INTEGER NV`	dimension of $v(t)$
`INTEGER NU`	dimension of $u(t)$
`INTEGER NP`	dimension of p
`INTEGER NRB`	dimension of r, i.e. number of boundary conditions
`DOUBLE PRECISION T0`	the initial time t_0
`DOUBLE PRECISION Y0(NY)`	values of $y(t_0)$
`DOUBLE PRECISION V0(NV)`	values of $v(t_0)$
`DOUBLE PRECISION U0(NU)`	values of $u(t_0)$
`DOUBLE PRECISION TF`	the final time t_f
`DOUBLE PRECISION YF(NY)`	values of $y(t_f)$
`DOUBLE PRECISION VF(NV)`	values of $v(t_f)$
`DOUBLE PRECISION UF(NU)`	values of $u(t_f)$
`DOUBLE PRECISION P(NP)`	values of p
`DOUBLE PRECISION RB(NY)`	resulting value of r ($\rightarrow$ Ausgabe/Output)

2.4 `MESSH` for the computation of the measurement functions h

function $h(t, y(t), v(t), u(t), p)$ of Eq. (4)

syntax `SUBROUTINE MESSH (NY,NV,NU,NP,NH,T,Y,V,U,P,H,NEEDH)`

parameters `IMPLICIT NONE`

`INTEGER NY`	dimension of $y(t)$
`INTEGER NV`	dimension of $v(t)$
`INTEGER NU`	dimension of $u(t)$
`INTEGER NP`	dimension of p
`INTEGER NH`	dimension of h
`DOUBLE PRECISION T`	time t
`DOUBLE PRECISION Y(NY)`	values of $y(t)$
`DOUBLE PRECISION V(NV)`	values of $v(t)$
`DOUBLE PRECISION U(NU)`	values of $u(t)$
`DOUBLE PRECISION P(NP)`	values of p
`DOUBLE PRECISION H(NH)`	resulting values of h ($\rightarrow$ Ausgabe/Output)
`INTEGER NEEDH(NH)`	array specifying which components of h have to be set on exit of the subroutine (if `NEEDH(`i`) > 0 then H(`i`)`, the i-th component of h, has to be set on exit).

2.5 `ZSTDB` for the computation of the (nonlinear) inequality constraints S

function $S(t, y(t), v(t), u(t), p)$ of Eq. (3)

syntax `SUBROUTINE ZSTDB (NY,NV,NU,NP,NZB,T,Y,V,U,P,ZB)`

parameters `IMPLICIT NONE`

`INTEGER NY`	dimension of $y(t)$
`INTEGER NV`	dimension of $v(t)$
`INTEGER NU`	dimension of $u(t)$
`INTEGER NP`	dimension of p
`INTEGER NZB`	dimension of S
`DOUBLE PRECISION T`	time t
`DOUBLE PRECISION Y(NY)`	values of $y(t)$
`DOUBLE PRECISION V(NV)`	values of $v(t)$
`DOUBLE PRECISION U(NU)`	values of $u(t)$
`DOUBLE PRECISION P(NP)`	values of p
`DOUBLE PRECISION ZB(NZB)`	resulting values of S ($\rightarrow$ Ausgabe/Output)

2.6 `ZLFKT` for the computation of the optimal control objective Φ

function $\Phi(t_f, y(t_f), v(t_f), u(t_f), p)$ of Eq. (6)

syntax `SUBROUTINE ZLFKT (NY,NU,NS,NP,TF,YF,VF,UF,P,PHI)`

parameters `IMPLICIT NONE`

`INTEGER NY`	dimension of $y(t)$
`INTEGER NV`	dimension of $v(t)$
`INTEGER NU`	dimension of $u(t)$
`INTEGER NP`	dimension of p
`DOUBLE PRECISION TF`	the final time t_f
`DOUBLE PRECISION YF(NY)`	value of $y(t_f)$
`DOUBLE PRECISION VF(NV)`	value of $v(t_f)$
`DOUBLE PRECISION UF(NU)`	value of $u(t_f)$
`DOUBLE PRECISION P(NP)`	value of p
`DOUBLE PRECISION PHI`	resulting value of Φ ($\rightarrow$ Ausgabe/Output)

3 Input File of ParEst

When starting ParEst specific data of the problem (dimensions, time grids, initial estimates, tolerances of optimization and numerical integration, selection between different integration and optimization methods, ...) are read from an input file (in the prepared example problems the input file is called **start.dat**).

Comments in the file **start.dat** always have to start with a # sign. If a # sign is found in a line of the input file, then the rest of the line will be ignored by ParEst.

The input file consists of several sections listed here. They consist of

`GENERAL`

dimensions, selection of integration and optimization methods, tolerances, ...

`ENDE`

`NAMEN`

names of the state and control variables (only to be used in the graphical output produced by PGraph*)*

`ENDE`

`MESSWERTE`

measurement values (only in the case of parameter estimation problems)

`ENDE`

`GRENZEN`

optional lower and upper bounds for the state and control variables $y(t)$, $v(t)$, $u(t)$, *and for the nonlinear inequality constraints* S

`ENDE`

`SKALIERUNG`

constants defining linear transformations for an (optional) internal scaling of the state and control variables $y(t)$, $v(t)$, $u(t)$ *and of the objective* Φ

`ENDE`

`STARTWERTE`

the (estimated) values of initial and final time and (estimated) values of the state variables at the **normalized** *multiple shooting grid (i. e. the normalized values of the multiple shooting nodes are within the interval* $[0, 1]$, *where 0 corresponds to* t_0 *and 1 corresponds to* t_f*)*

`ENDE`

`STEUERUNG`

estimated values of the control variables at the **normalized** *control grid points (i. e. the normalized values of the control discretization grid points are within* $[0, 1]$*, where 0 corresponds to* t_0 *and 1 corresponds to* t_f*)*

`ENDE`

The sections are described in more detail in the sequel. Each section contains statements of the syntactic form

name of variable (or dimension) = value

For example:

```
DIMY = 3
```

For more details see the programs of the prepared examples.

3.1 Section GENERAL

This section contains the description of the dimensions of the problem and of optional parameters of the optimization method. They are set by assigning values to the following variable names:

`DIMY`	dimension NY of the differential state variable $y(t)$ (always has to be greater or equal one!).
`DIMV`	dimension NV of the algebraic state variable $v(t)$ (in the case of only ordinary differential equations: `DIMV` $= 0$).
`DIMU`	dimension NU of the control variable $u(t)$ (may be zero as it is usually the case in parameter identification problems).
`DIMH`	dimension NH of the function $h(t, y(t), v(t), u(t), p)$ (`DIMH` $= 0$ is possible).
`NPAR`	number NP of control parameters p (`NPAR = 0` is possible).
`NRB`	number of boundary conditions $r(t_0, y_0, v_0, u_0, t_f, y_f, v_f, u_f, p) = 0$ (usually `NRB` $= 0$ will not be useful but might be appropriate for some parameter identification problems).
`NZB`	number NZB of nonlinear inequality constraints $a_i \leq S_i(t, y(t), v(t), u(t), p) \leq b_i$.

The selection between parameter identification and optimal control problems is done by setting

`ZIELFKT`	which specifies the objective: `ZIELFKT` = 0 – for minimizing a nonlinear least squares objective (→ parameter identification problems) `ZIELFKT` = 1 – for minimizing an objective of Mayer type (→ optimal control problems)

Free or prescribed initial and final time are set by using the statements

`FEST_E(1)`	for a fixed initial time t_0,
`FREI_E(1)`	for a free initial time t_0,
`FEST_E(2)`	for a fixed final time t_f,
`FREI_E(2)`	for a free final time t_f

in this section of the input file.

Three different methods can be selected for the numerical integration of the dynamical equations (1)

1. DASSL: the initial-value problem solver for differential-algebraic equations of [13] (backward difference formulas) with the extensions of [1] for a numerical sensitivity analysis.
2. DOPRI8: the initial-value problem solver for ordinary differential equations of [10] (8th order extrapolation method) with the extensions of [6] for a numerical sensitivity analysis.
3. RKF4: an initial-value problem solver for ordinary differential equations (4th order Runge/Kutta method) with the extensions of [6] for a numerical sensitivity analysis.

`INTGRT`	Selection of the integration method `INTGRT` = 1 – DASSL, `INTGRT` = 2 – DOPRI8, `INTGRT` = 3 – RKF4.
`RTOL`	for DASSL: relative tolerance of integration for DOPRI8: accuracy of integration for RKF4: accuracy of integration
`ATOL`	for DASSL: absolute tolerance of integration for DOPRI8: accuracy for the computation of the sensitivity matrices for RKF4: not used

As a rule of thumb, the tolerances for integration should be less or equal one tenth of the tolerances for optimization (see below), for example, RTOL $= 10^{-6}$ and OPTTOL $= 10^{-5}$ (see description below).

Three iterative methods for optimization are available:

1. NLSCON (Nowak, Weimann [11], [12]):
   a generalized Gauß-Newton method for nonlinear least squares problems subject to nonlinear equality constraints [8]
   (NLSCON can **only** be used for parameter identification problems and **not** for optimal control problems),

2. NPSOL (Gill, Murray, Saunders, Wright [9]):
   a Sequential Quadratic Programming (SQP) method for general nonlinearly constrained optimization problems [14]
   (NPSOL is the **only** possible selection for optimal control problems. It may also be selected for parameter identification problems but will usually be less efficient than NLSSOL or NLSCON),

3. NLSSOL (Gill, Murray, Saunders, Wright [9]):
   an SQP-method for nonlinear least squares problems subject to general nonlinear equality and inequality constraints
   (NLSSOL can **only** be used for parameter identification problems and **not** for optimal control problems).

`OPTFCN`	selects the optimization method `OPTFCN` = 1 – NLSCON `OPTFCN` = 2 – NPSOL `OPTFCN` = 3 – NLSSOL
`OPTTOL`	optimality tolerance for optimization
`NLFTOL`	feasibility tolerance of the nonlinear constraints at a solution (only used by NPSOL and NLSSOL)
`PRINTLVL`	print level of iterations (amount of output for each iteration) for example, `PRINTLVL` = 0 for no output or, for some brief output, `PRINTLVL` = 3 for NLSCON and `PRINTLVL` = 5 for NPSOL and NLSSOL
`NSTEPS`	(optional) maximum number of iteration steps of the optimization method (if NSTEPS is not specified, a reasonable value will be selected internally) **Remark:** During iterations the program can also be interrupted interactively by typing <`Ctrl`>-`c`.

	In this case, ParEst prints the message:
	`Execution will be terminated`
	`Please wait ...`
	and the program will terminate after the current iteration has been completed and save the current estimate of the solution in the output file **solution.dat**.

3.2 Section NAMEN

In this section, descriptions of variables and functions can be supplied for use as descriptions in the graphical output prepared by PGraph.

The syntax for setting descriptions is

$$variable = \text{¨}description\text{¨}$$

As *variable*

`Y(`i`)`	for the i-th differential state variable $y_i(t)$
`V(`i`)`	for the i-th algebraic state variable $v_i(t)$
`U(`i`)`	for the i-th control variable $u_i(t)$
`H(`i`)`	for the i-th function $h_i(t, y(t), v(t), u(t), p)$

can be used.

Example:

```
Y(1) = ¨x(t)¨
```

3.3 Section MESSWERTE

This section is used only in the case of a parameter identification problem in order to provide the program with the measurement values z_{ij} appearing in Equation (5). For each measurement value z_{ij}, also a positive, real-valued weight s_{ij} can optionally be supplied.

The measurement values have to be listed in the **chronological order** of the measurement times in this section of the input file.

Measurement values at time t of measurement have to be provided by

`T =` *time t of the measurement*
`[Y(`i`) =` *measured value of* y_i`(T) [SY(`i`) =` *corresponding positive weight*`]]`
`[V(`i`) =` *measured value of* v_i`(T) [SV(`i`) =` *corresponding positive weight*`]]`
`[H(`i`) =` *measured value of* h_i`(T) [SH(`i`) =` *corresponding positive weight*`]]`

The terms in brackets "[$\cdots$]" may be supplied optionally but don't have to be supplied always.

In many applications, the weights s_{ij} are related to the standard deviations of the measurements and are equal to one constant value for the i-th variable at any times. In this case a standard weight can be set to the i-th variable by writing in this section **before** listing the measurement values

`[SY(`i`) =` *value of the standard weight of the measured i-th differential state variable* $y_i(t)$`]`
`[SV(`i`) =` *value of the standard weight of the measured i-th algebraic state variable* $v_i(t)$`]`
`[SH(`i`) =` *value of the standard weight of the measured i-th function* $h_i(t)$`]`

`T =` ... (values of the first measurement).

3.4 Section GRENZEN

In this section, lower and upper bounds (i. e., box constraints) can be set for the state and control variables, the control parameters, the initial and final time, and for the nonlinear inequality constraints S from Equation (3).

This option **only** works with NPSOL and NLSSOL but **not** with NLSCON.

The syntax for setting bounds is

variable = *value of the lower bound* *value of the upper bound*

If an upper bound is unconstrained then a value greater or equal to $+10^{10} \equiv +1.E{+}10$ has to be set.
Or, if a lower bound is unconstrained then a value less or equal to $-10^{10} \equiv -1.E{+}10$ has to be set.

As *variable* one may use:

`E(1)`	for bounds of the initial time t_0.
`E(2)`	for bounds of the final time t_f.
`P(`i`)`	for bounds of the i-th control parameter p_i.
`Y(`i`)`	for bounds of the i-th differential state variable $y_i(t)$.
`V(`i`)`	for bounds of the i-th algebraic state variable $v_i(t)$. **Please note that** the bounds on $y_i(t)$ (and on $v_i(t)$ as well) provided in this section will be satisfied only at the multiple shooting nodes t_j^m. State variable inequality constraints of the type $a \leq y_i(t) \leq b$, for all $t \in [t_0, t_f]$, have to be specified as an inequality constraint function S.
`U(`i`)`	for bounds of the i-th control variable $u_i(t)$. **Please note that** the bounds on $u_i(t)$ provided in this section will be satisfied at the grid points t_j^s of the control discretization grid. Because of the linear approximation of the control variables the computed piecewise linear control approximation will satisfy this bounds **everywhere** in $[t_0, t_f]$.
`ZB(`i`)`	for bounds a_i, b_i of the i-th inequality constraint function $S_i(t, y, v, u, p)$ of Eq. (3). **Please note that** these inequality constraints will be satisfied only at the control discretization grid points t_j^s.

3.5 Section SKALIERUNG

A proper scaling of all dependent and independent variables and functions may effect the efficiency and robustness of any numerical algorithm used on computers with limited arithmetic precision.

It is therefore recommendable to scale the state variables $y(t)$ and $v(t)$, the control variables $u(t)$ and also the objective function Φ properly by applying suitable linear transformations.

As a rule of thumb, the range of the transformed variables $\bar{y}_i(t)$, $\bar{v}_i(t)$ and $\bar{u}_i(t)$ should be approximately $[-1, +1]$ (after scaling). If the original range of the variables is not known a priori one may get a first idea by looking at the initial trajectories.

For the purpose of scaling by linear transformation the formulas

$$\begin{aligned} \bar{y}_i(t) &= (y_i(t) - B_i^y)/S_i^y\,, \quad B_i^y \in \mathbb{R}\,, S_i^y \in \mathbb{R}^+\,, \quad i = 1, \ldots, NY, \\ \bar{v}_i(t) &= (v_i(t) - B_i^v)/S_i^v\,, \quad B_i^v \in \mathbb{R}\,, S_i^v \in \mathbb{R}^+\,, \quad i = 1, \ldots, NV, \\ \bar{u}_i(t) &= (u_i(t) - B_i^u)/S_i^u\,, \quad B_i^u \in \mathbb{R}\,, S_i^u \in \mathbb{R}^+\,, \quad i = 1, \ldots, NU, \end{aligned} \tag{10}$$

are used. The constants B_i^y, B_i^v, B_i^u, S_i^y, S_i^v, and S_i^u have to be selected properly in order to transform the ranges of the state variables and the control variables onto $[-1, +1]$ (as a rule of thumb).

In the same manner, the objective can be transformed by

$$\bar{\Phi} = (\Phi - B^{\Phi})/S^{\Phi}\,, \quad B^{\Phi} \in \mathbb{R}\,, S^{\Phi} \in \mathbb{R}^{+}. \tag{11}$$

As a rule of thumb, the minimum value of the transformed objective should be approximately 1000 and the range of variation of the transformed objective $\bar{\Phi}$ should be between zero and 1000.

The scalings by linear transformations are optionally. The constants defining the linear transformations can be provided in this section using the syntax

$$\textit{variable} = \textit{value of } B_i \quad \textit{value of factor } S_i$$

As *variable* one may use

`P(`i`)`	for scaling of the i-th control parameter p_i
`Y(`i`)`	for scaling of the i-th differential state variable $y_i(t)$
`V(`i`)`	for scaling of the i-th algebraic state variable $v_i(t)$
`U(`i`)`	for scaling of the i-th control variable $u_i(t)$
`ZIELFKT`	for scaling of the objective Φ

The linear transformations are then applied **internally**. The user **does not** have to deal with the transformed variables. Input, output and problem dependent subroutines are handled in the original units of all variables and functions.

Example: Looking at the initial trajectory of an optimal control problem from flight mechanics one may find that that the differential state variable $y_1(t)$ (altitude) does have a range of approximately 30×10^3 (meters) to 450×10^3 (meters). In this case, it may be useful to select $B_1^y = 240 \times 10^3$ and $S_1^y = 210 \times 10^3$.

These constants for the internal linear transformation of $y_1(t)$ are now supplied to the program by the line

```
Y( 1) = 240.0E+3 210.0E+3
```

in section SKALIERUNG of the input file.

3.6 Section STARTWERTE

In this section, first the values of initial time and final time have to be provided by

`E(1) =` *value of the initial time* t_0
`E(2) =` *value of the final time* t_f

If either initial or final time are free and not fixed (see Section GENERAL of the input file) then an estimated value has to be used here.

Next the initial estimates of the control parameters p_i have to be supplied (if there are any) by

`P(`i`) =` *estimate of the value of the i-th control parameter* p_i

Then a list of the multiple shooting nodes t_j^m follows where each multiple shooting node is accompanied by a list of estimates of the values of the state variables $y_i(t_j^m)$ and $v_i(t_j^m)$ at the multiple shooting nodes.

The list of multiple shooting nodes **must** have **at least two** entries, namely for the initial time and for the final time.

The (constant) values of the multiple shooting nodes can be provided in two different ways:

- By using the absolute value t where $t_0 \leq t \leq t_f$

  `T =` *absolute value of the multiple shooting node*

- or by using the corresponding normalized value $\tau \in [0, 1]$ (where $t = \tau \cdot (t_f - t_0) + t_0$)

  `TAU =` *normalized value of the multiple shooting node*

Example: Consider the case of a time period of $[t_0, t_f] = [0, 5]$. A multiple shooting node $t = t_k^m = 2$ (absolute value) can now be selected either by

```
T = 2.0
```

or by the normalized value $\tau = \tau_k^m = (t_k^m - t_0)/(t_f - t_0) = 0.4$

```
TAU = 0.4
```

A list of estimates of the values of the state variables has to accompany each multiple shooting node t_j^m in the form of

`Y( 1) =` *estimated value of* $y_1(t_j)$

$\vdots$

`Y(NY) =` *estimated value of* $y_{NY}(t_j)$

`V( 1) =` *estimated value of* $v_1(t_j)$

$\vdots$

`V(NV) =` *estimated value of* $v_{NV}(t_j)$

In the case of a **free final time** t_f it is recommended to select multiple shooting nodes by normalized values. Then their **relative** position in $[t_0, t_f]$ will be constant for any value of t_f while their absolute values will vary depending on the current value of t_f.

Remarks:

- If the functions of the right hand side of the differential-algebraic equations are only **piecewise** differentiable (for example, if they are piecewise defined), it is absolutely **necessary** to place a multiple shooting node at such a matching point of the functions of the right hand side. Otherwise uncontrollable errors may occur in the integration method and in the computation of the gradients that will cause the method to fail.
- **Please note** that the **first** and **last** multiple shooting nodes **always** have to be equal to initial time and final time.

3.7 Section STEUERUNG

In this section the control discretization grid points t_j^s and also estimated values of the control variables $u_i(t_j^s)$ at the grid points have to be provided.

The grid points of the control discretization grid can be selected by absolute or normalized values in the same manner as described for the multiple shooting nodes in the section above.

`T =` *absolute value of the control discretization grid point* t_j^s

or

`TAU =` *normalized value of the control discretization grid point* t_j^s

Each grid point t_j^s of the control discretization has to be accompanied by a list of estimated values of the control variables $u_i(t_j^s)$ at the grid point

`U( 1) =` *estimated value of* $u_1(t_j^s)$

$\vdots$

`U(NU) =` *estimated value of* $u_{NU}(t_j^s)$

Remarks:

- The multiple shooting grid (t_j^m) and the control discretization grid (t_j^s) can be selected independently.
- **Please note** that the **first** and **last** control discretization grid points **always** have to be equal to initial and final time.
- If the functions of the right hand side of the differential-algebraic equations are only **piecewise** differentiable we recommend to place a control grid point at such a matching point of the model functions.

4 The Programs `parest` and `pgraph`

To investigate a new parameter identification or optimal control problem using ParEst we recommend to get a copy of all files of one of the examples and modify them for the new problem.

First, the file **uspec.f** of the problem dependent subroutines (described in Chapter 2) and the input file **start.dat** (described in Chapter 3) have to be prepared.

Next, the programs `parest` and `pgraph` can be compiled and linked by simply typing `make` (which makes use of the prepared `Makefile`).

parest The optimization is done by program `parest`. It is started by typing

```
parest
```

if the name of the input file is `start.dat` and the name of the output file is `solution.dat`. If input and/or output files do have other names they might be used as well. The syntax is

`parest` [*input file* [*output file*]]

The terms in brackets "[···]" may be supplied optionally but don't have to be supplied always.

The default name of the **output file** is `solution.dat`. The output file contains a brief information about the optimization and two sections, namely STARTWERTE and STEUERUNG. These values can now be used as improved estimates of the state variables at the multiple shooting nodes (Section STARTWERTE of the input file) and of control variables at the control discretization grid points (Section STEUERUNG of the input file) for a subsequent run of ParEst if the corresponding sections of the input file are replaced by these values.

A run of ParEst can also be terminated interactively at any time by typing `<Ctrl>-c` (or, of course, using the UNIX command `kill` on a UNIX machine). Then the program will respond by printing

```
Execution will be terminated
Please wait ...
```

The program will terminate immediately after the current iteration has been completed and save the current estimate of the solution in the output file **solution.dat**.

pgraph A graphical output is prepared by the post-processing program PGRAPH. It is based on the graphical system of the LRZ [3].

The program can be used with the following options

pgraph [-l] [-g] [-k] [-s] [-P *number*$_1$/*number*$_2$]
[-n *number of figures per page*]
[-p *number of approx. points for curves*] [-h step size]
[-d *input file*] *solution file*$_1$ [... *solution file*$_n$]
[#*list*$_1$ [... #*list*$_m$]]

Description of options:

l Tables of function values will be generated as output of PGRAPH which may be used for visualization with another software than PGRAPH.
The tables are in files named `pgraph#`xxx`.dat` where xxx is just the number of the i-th figure.

g A dotted rectangular coordinate grid covering the whole figure will **not** be drawn.

k The location of each multiple shooting node will be marked by a small circle $\circ$ on each curve.

s The location of each control discretization grid point will be marked by a square $\diamond$.

P i/j A figure of a parameterized curve (phase diagrams) will be generated by using the values of the i-th curve on the horizontal achsis and of the j-th curve on the vertical achsis.
The numbers i and j correspond to the following variables and functions

	i (or j)
y_i	$1, \ldots, NY$
v_{i-NY}	$NY+1, \ldots, NY+NV$
$u_{i-NY-NV}$	$NY+NV+1, \ldots, NY+NV+NU$
$S_{i-NY-NV-NU}$	$NY+NV+NU+1, \ldots,$ $NY+NV+NU+NZB$
$h_{i-NY-NV-NU-NZB}$	$NY+NV+NU+NZB+1, \ldots,$ $NY+NV+NU+NZB+NH$

n i Up to $i = 4$ figures may be collected on one sheet of the grahical output file. The default value is $i = 1$.

p n All curves will be plotted by linear interpolation of n equidistant approximation points in $[t_0, t_f]$. The default value is $n = 100$. A larger value of n will result in smoother looking curves, but will also increase computational time and storage requirements for the resulting graphics file.

h s An alternative option to **-p**. For plotting the curves, equidistant approximation points will be used. The equidistance is $\Delta t = s$ of (absolute) time units. A smaller value of s will result in smoother looking curves, but will also increase computational time and storage requirements for the resulting graphics file.

d *file* Name of the input file. The default value is `start.dat`.

Curves corresponding to different runs of ParEst can be drawn in the same figure by simply listing the names of the corresponding output files *solution file*$_1$,..., *solution file*$_n$.

Also reference values from other source can be drawn in the same figure as the solution of ParEst. They have to be provided in form of tables of function values. These files have to be distinguished from solutions files of ParEst by typing "'#"' as an initial to their file names.

The format of such tables with reference values is:

number of function values *number of functions*

curve number$_{column_0}$ ... *curve number*$_{column_N}$

$value_0$ ... $value_N$

$\vdots$

$value_0$ ... $value_N$

The numbers of curves are interpreted in a way described for option **-P**. If one of the columns of the table may not be drawn then the curve number -1 has to be used.

Examples for calls of PGraph:

1. `pgraph -k -s -p500 solution.dat`
   A graphical file is generated using 500 equidistant points for drawing curves. Multiple shooting nodes and control discretization grid points will be marked.

2. `pgraph -l -h 0.001 solution.dat`
   Tables of functions will be generated in addition to the graphical output file. For the tables and for the drawing curves equidistant points are used at any 0.001 times.

3. `pgraph solution.dat '#refsol.dat'`
   In addition to the curves from the outputfile `solution.dat` curves with values from the file `refsol.dat` will be plotted.

For example, consider a problem with the dimensions NY = DIMY = 2, NV = DIMV = 0, NU = DIMU = 0, NZB = 1, NH = DIMH = 1.

The first two rows of the file with 100 reference values for 5 variables or functions in tables to be drawn in comparison with the solutions of ParEst may read as

100 5
0 1 3 -1 2 4

Now the interpretation of the columns of the tabular values which are following these two rows is

| t | $y_1(t)$ | $s_1(t)$ | will not be drawn | $y_2(t)$ | $h_1(t)$ |

As standard output, `pgraph` generates an output file named **ZZZG67** which is prepared for previewing by the command `graphik` of [3].

Examples for calls of `graphik`:

1. `graphik -n xprev -c -z ZZZG67`
   For previewing the graphics at the display.

2. `graphik -n ps -c -z ZZZG67`
   For converting the whole file ZZZG67 to a PostScript-file ZZZG67.ps. By the `-c` option the output is in color if sent to a color printer. No previewing of the graphics is done at the display.

Refinement of the control functions
The function tables generated by the **–l** option of PGraph can be used to obtain a finer approximation of the control variables u.

For example we have a solution of an optimization problem where the control function has been calculated at 10 equidistant points. In the next optimization run we like to refine the control and calculate its values at 20 equidistant points. The following run of PGraph generates a function table of all variables in the file `pgraph#00.gdat` at the desired locations:
`pgraph -l -p 20 solution.dat`

Assume we have three state variables and one control function, then the function table contains five columns. The first column holds the time, the following three the corresponding values of the state variables and the fifth the values of the control variable.

The following **awk** script extracts the values of the control function out of the file `pgraph#00.gdat` and prints a new section for the control:

```
BEGIN   { print "STEUERUNG" }

/^[^#]/ { # Each line not beginning with '#'
          print "   T =   ", $1
          print "      U(1) = ", $5
        }

END     { print "ENDE" }
```

If we name the script `genu.awk` the new section is generated by typing
`awk -f genu.awk < 'pgraph#00.gdat' > newu.dat`

Finally we have to replace the section **STEUERUNG** in the file `solution.dat` by the new one in `newu.dat`. We now can invoke the optimization with
`parest start.dat solution.dat`

5 Examples

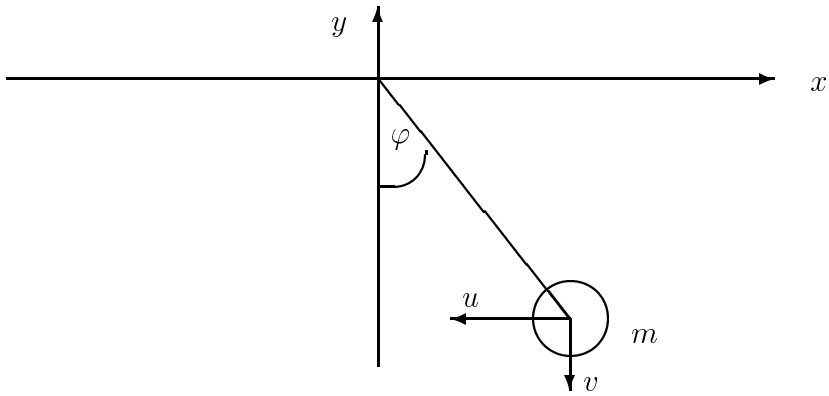
For the two examples in this chapter, the problem dependent files **uspec.f** and **start.dat** are already prepared in the subdirectories `pendulum` and `minenergy` of the directory of the examples.

The underlying mathematical formulation of both problems is described briefly in the sequel.

The two problems have been selected for the purpose of demonstration. More sophisticated problems and results obtained using ParEst can be found in [1] and [2].

5.1 Parameter identification of a planar pendulum

Consider a pendulum of length l and mass m which is fixed at the origin of a coordinate system and moves in the x-y plane.



We consider the formulation of the dynamics of motion as a system of differential-algebraic equations. If λ denotes the Lagrangian multiplier and g the gravitational constant then the Lagrangian is

$$L = \frac{m}{2}(\dot{x}^2 + \dot{y}^2) - mgy + \lambda(x^2 + y^2 - l^2)\,. \tag{12}$$

Now substituting $\dot{x} =: u$, $\dot{y} =: v$ for the velocities and applying index reduction techniques, the equations of motions read as a system of differential-algebraic equations of index 1

$$\begin{aligned} \dot{x}(t) &= u(t) \\ \dot{y}(t) &= v(t) \\ \dot{u}(t) &= \lambda x(t)/m \\ \dot{v}(t) &= \lambda y(t)/m - g \\ 0 &= u^2(t) + v^2(t) + \lambda(t) l^2/m - y(t) g\,. \end{aligned} \tag{13}$$

Whereas the two additional entry conditions to be satisfied at t_0 due to the index reduction are

$$\begin{aligned} 0 &= x^2(t_0) + y^2(t_0) - l^2 \\ 0 &= x(t_0)u(t_0) + y(t_0)v(t_0). \end{aligned} \tag{14}$$

Now the dynamic model has the dimensions

$NY = \text{DIMY} = 4$, $NV = \text{DIMV} = 1$, $NU = \text{DIMU} = 0$, $NP = \text{NPAR} = 1$, $NRB = \text{NRB} = 2$, $NZB = \text{NZB} = 0$.

We assume that the gravitational constant g is unknown and has to be identified by using the results of an experiment:
The pendulum moves for two seconds. After every 0.2 seconds the values of $x(t)$, $y(t)$, and $\lambda(t)$ are measured (but the values of the velocities $u(t)$, $v(t)$ are not). This gives 11 times t_1^e,...,t_{11}^e of measurements and 33 measurement values in total.

It is assumed that the measurements are not fully precise but with measurement errors of standard deviations of $\sigma_x = \sigma_y = 0.05$ and $\sigma_\lambda = 0.25$. These weights will be used as weights s_{ij} in the nonlinear least squares objective of Equation (5).

A multiple shooting node is selected at every time of measurement. As initial estimates of the state variables $x(t)$, $y(t)$ and $\lambda(t)$ at the multiple shooting nodes we simply use the given measurements. Initial estimates of the velocities $u(t)$, $v(t)$ can be computed by local interpolation of the measurements of $x(t)$, $y(t)$.

As initial estimate of the parameter g we use $20\,\mathrm{m/s^2}$.

The "true" value of g is $9.81\,\mathrm{m/s^2}$.

5.2 A minimum energy problem

This well-known problem is reported in Sec. 3.11, Example 2, of Bryson, Ho [5]. The task is to minimize

$$J[u] = \frac{1}{2}\int_0^1 u^2(t)\,dt \longrightarrow \min! \tag{15}$$

subject to the differential equations and boundary conditions

$$\begin{aligned} \dot{x}(t) &= v(t)\,, & x(0) &= 0\,, & x(1) &= 0 \\ \dot{v}(t) &= u(t)\,, & v(0) &= 1\,, & v(1) &= -1 \end{aligned} \tag{16}$$

and subject to the state variable inequality constraint

$$x(t) \le l\,, \quad l = \text{const} \in \mathbb{R}^+\,. \tag{17}$$

First the objective has to be transformed into Mayer type by introducing a third state variable

$$\dot{w}(t) = \frac{1}{2}u^2(t), \quad w(0) = 0\,. \tag{18}$$

Then the objective to be minimized is

$$J[u] = w(1) \longrightarrow \min! \tag{19}$$

Now the dynamic model has the dimensions

NY = DIMY = 3, NV = DIMV = 0, NU = DIMU = 1, NP = NPAR = 0, NRB = NRB = 5, NZB = NZB = 1.

A formula of the solution can be given explicitly (see [5] for details). For $l > 1/4$ the inequality constraint does not become active, for $1/6 \leq l \leq 1/4$ the inequality constraint becomes active at a touch point at $t = 0.5$, and for $0 \leq l < 1/6$ the inequality constraint becomes active a along a whole subarc of $[0, 1]$.

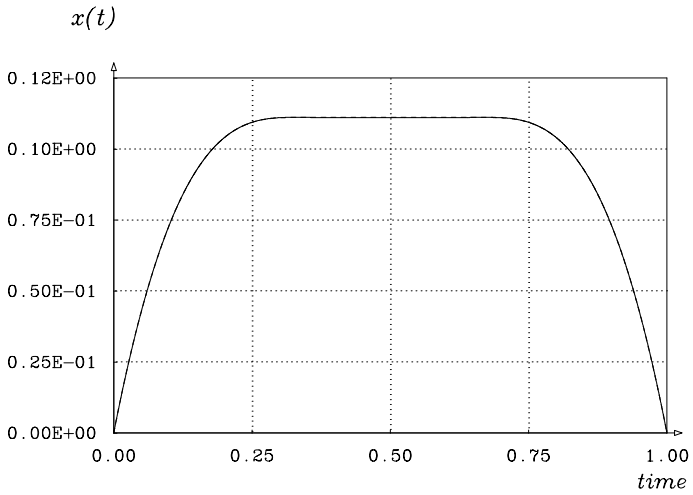


Figure: Optimal $x(t)$ for an upper bound of $l = 1/9 = 0.1111\ldots$

The unconstrained problem (`NZB = 0`) can be investigated first. Next, the state constrained problem can be investigated by setting `NZB = 1` and l properly (for example, $l = 1/6$ or $1/9$). The behaviour of the solution may now be studied for a sequence of increased or decreased values of l.

References

[1] Heim, A.:
Parameteridentifizierung in differential-algebraischen Gleichungssystemen.
Diploma thesis (München: Technische Universität, Mathematisches Institut, 1992).

[2] Heim, A.; von Stryk, O.:
A direct multiple shooting method for parameter estimation and optimal control in systems of differential-algebraic and ordinary differential equations. Part 1: Description of the method, Part 2: Numerical results. In preparation.

[3] Apostolescu, V.; Weidner, K.:
Das LRZ-Graphiksystem. Teil III: "Previewing" am Sichtgerät.
Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, LRZ-Benutzerschrift Nr. 7904/5, 5. Aufl. (München: Jan. 1986).

[4] Bock, H. G.; Eich, E.; Schlöder, J. P.:
Numerical solution of constrained least squares boundary value problems in differential-algebraic equations.
In: K. Strehmel (ed.): Numerical Treatment of Differential Equations. Prof. NUMDIFF-4 Conference, Halle-Wittenberg, Germany, 1987, Teubner Texte zur Mathematik **4** (Teubner, 1988) pp. 269–280.

[5] Bryson, A. E.; Ho, Y.-C.:
Applied Optimal Control.
Rev. Printing. (Hemisphere Publishing Corporation, New York, 1975).

[6] Buchauer, O.; Hiltmann, P.; Kiehl, M.:
Sensitivity analysis of initial-value problems with application to shooting techniques.
Numer. Math. **7** (1994) pp. 151–159.

[7] Bulirsch, R.:
Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung.
DLR, Oberpfaffenhofen, Report of the Carl-Cranz-Gesellschaft (1971). Reprints: Mathematisches Institut, Technische Universität München (1985, 1993).

[8] Deuflhard, P.; Apostolescu, V.:
An underrelaxed Gauß-Newton method for equality constrained nonlinear least squares problems.
In: J. Stoer (ed.): Optimization Techniques. Lecture Notes in Control and Information Sciences **7** (Berlin: Springer, 1978) pp. 22–32.

[9] Gill, P.E.; W. Murray; M.A. Saunders; M.H. Wright:
User's Guide for NPSOL (Version 4.0).
Report SOL 86-2. Department of Operations Research, Stanford University, California (1986).
User's Guide for NPSOL (Version 5.0).
Report SOL 94-?. Department of Operations Research, Stanford University, California, to appear.

[10] Hairer, E.; Wanner, G.; Norsett, S.P.:
Solving Ordinary Differential Equations I.
Springer Series in Computational Mathematics **8**, 2nd ed. (Springer, 1993).

[11] Nowak, U.; Weimann, L.:
NLSCON: Numerical solution of nonlinear (NL) least squares (S) problems with nonlinear constraints (CON).
Version 2.3 (Fortran 77). Konrad Zuse Zentrum für Informationstechnik, Berlin (1992).

[12] Nowak, U.; Weimann, L.:
A Family of Newton Codes for Systems of Highly Nonlinear Equations — Algorithm, Implementation, Application.
ZIB, Konrad Zuse Zentrum für Informationstechnik, Berlin, Technical Report TR 90-10 (1990).

[13] Petzold, L.:
A description of DASSL: A differential/algebraic system solver.
SAND82-8637, Sandia National Laboratories (Sep. 1982).

[14] Schittkowski, K.:
The Nonlinear Programming Method of Wilson, Han, and Powell with an Augmented Lagrangian Type Line Search Function.
Part 1: Convergence Analysis.
Part 2: An Efficient Implementation with Linear Least Squares Subproblems.
Numer. Math. **38** (1981) pp. 83–127.

[15] Stoer, J.; Bulirsch, R.:
Introduction to Numerical Analysis.
2nd ed. (Springer, 1993) 660 p.