Standards in Computer Aided Analysis and Synthesis Of Mechanisms

Antonius J. KLEIN BRETELER
Delft University of Technology, Faculty 3ME, Transport&Logistics
Mekelweg 2, 2628CD Delft, The Netherlands
a.j.kleinbreteler@tudelft.nl

1 ABSTRACT

The paper discusses the need for a standard of describing (modeling) a mechanism. Automatic conversion of a mechanism subject to all design theories is the final goal. To start the discussion the author describes his experience with a specific theory (FEM) and the mechanism model used for it. A second standardization topic concerns the functions used for input or output with the design theories, regarding data exchange between various computer programs. The author describes his attempts and results regarding kinematic and dynamic analysis of planar mechanisms. It can be concluded that these results are a first setup for such a standard.

Keywords: mechanism model, function model, model conversion, data exchange, Runmec, PION

2 INTRODUCTION

In the past decades various computer programs have been developed to assist in designing mechanisms. Some of them concern a specific calculation task in a case study; others have the intention to be a general computer tool that could be used by other designers. Both at universities and in commercial companies software developments for mechanism design took place.
The intention of this paper is to discuss the necessity for standards in mechanism design, as required in the early design stages (kinematic, dynamic) analysis and synthesis of mechanisms. It is observed that various types of mechanism models are used like based on:

- Loop equations (vector sum);
- Assur Groups;
- (finite) Elements.

When using different computer programs in a certain case study, it can be required to convert the mechanism from one model to another model type. Developing generic conversions between all possible model types will lead to numerous conversion rules. It seems wise then to consider a “super model”, that should have a bi-directional conversion with all other models. That would reduce the number of possible conversions. Until now some serious proposals for such a super model have been made. In [2] a generic mechanism model has been proposed to assist for all mechanism design activities including manufacturing of parts. The proposal, with additional publications [3] and [4], describes a data structure that is typically suited for a database. Due to its ambitious intention the model appears to be rather complex. A more limited ambition is the scope of this paper, but it is still unclear yet if such a model can be made such that it is felt as a valuable product. But it is certainly desirable to discuss the super model as a candidate for a standard in mechanism design.
Except for the mental concept of a mechanism model, the implementation in a computer program can be regarded for a standard. By standardized data exchange it will become easier to use the most appropriate software during the design of the mechanism.
A second focus is laid on the various products that serve as input or output of mechanism design: functions, like

- (kinematic) Transfer function;
- Curve traced by a point;

- Timed motion behaviour;
- Cam profile (input);
- Spring characteristic (input);
- Required motion (input, synthesis).

Here a comparable question is how these products can easily be used in different computer programs [1]. It seems wise to have a standard for data exchange.

To serve the discussion of the above topic, the author presents in this paper his own experience with the two topics "mechanism model" and "function model". The mechanism model used in the program Runmec (FEM-based, developed by the author [5]) will be described in chapter 3. The various functions, described in chapter 4, underline the necessity to exchange data easily. The current state-of-the-art is the function model of the software called PION [6]. It will be presented in chapter 5 as a starting point for discussion and development of a standard function model.

3 MECHANISM MODEL OF RUNMEC (FEM, 2D)

3.1 Element definition

The most common element for linkages is the binary element, see fig. 1. The position of an element in the fixed world is defined by a set of co-ordinates (notation vector $\underline{x}$), which are here the x- and y-co-ordinates of the end-points. In general co-ordinates can be any geometrical quantity, like an angle to identify the orientation of an element. With respect to this concept the co-ordinate vector is considered as a generalized vector.

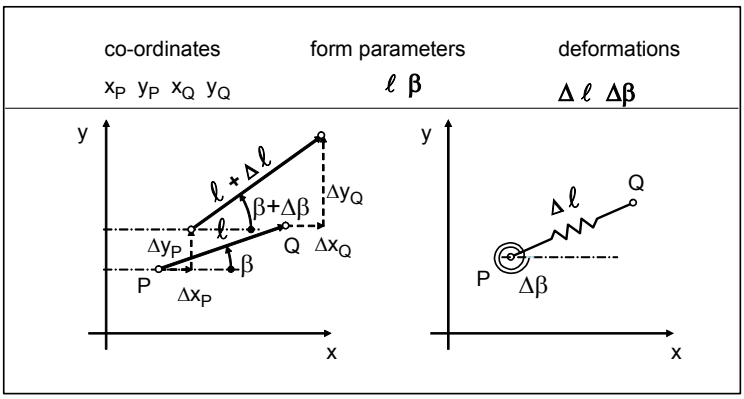

Fig. 1 Planar binary element

All motion properties of the element are calculable when the motion of the co-ordinates is known. So-called form parameters ε may be defined to specify a relation (continuity equation) between the co-ordinates. Any continuous function of the co-ordinates ε($\underline{x}$) can be regarded, but for practical use a clear physical meaning is wanted. For this element the parameters „length" and „angle" are useful. By user choice these parameters must be labeled either as prescribed (deformation zero in kinematics) or as dependent (free).

In this way various element subtypes (bar, prismatic pair) can be specified, see fig. 2. Characteristic for this FEM-modeling is that an element does not need to be identical with a body, but should be regarded as „a group of co-ordinates with possibly some relations between them". Because the continuity equations are in general non-linear, the solution of them requires linearization. The Jacobian matrix of the element, partial derivatives $\partial\varepsilon/\partial x$ of form parameters to co-ordinates, is needed. Fig. 3 shows this matrix for the binary element.

element type	deformations	kinematic scheme
truss	$\Delta\ell = 0$ $\Delta\beta$ = free	
slider in frame	$\Delta\ell$ = free $\Delta\beta = 0$	
prismatic pair	$\Delta\ell$ = free $\Delta\beta$ = free	
actuator, translating	ℓ = prescribed	
actuator, rotating	β = prescribed	

Fig. 2 Modelling kinematic structures with the binary FEM-element

3.2 Kinematic analysis

The mechanism has, by its assembled elements, a set of continuity equations. The central role is played by the set of first order equations, of which the matrix of „moving" co-ordinates x^c and prescribed form parameters ε^p is to be selected. The inverse of this assembled and selected Jacobian matrix D^{pc}, which in general exists for a square and regular matrix, contains all partial derivatives $\partial x^c / \partial \varepsilon^p$ of the mechanism (fig. 4). These include the (kinematic) transfer function of first order, since the driving quantities are modeled as prescribed form

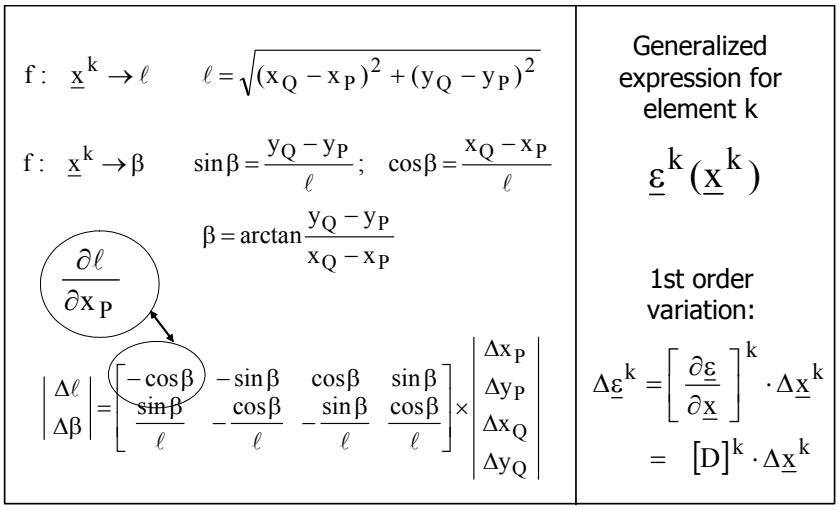

Fig. 3 Continuity equations of the binary element

parameters. The solution of this linear system of equations is to be regarded as „exact“, even when the result of matrix reduction is numerical.

Position analysis of the mechanism (solution of the nonlinear continuity equations) needs a numerical scheme like Newton-Rapson. Starting in a given position, the inputs are incremented and the solution is calculated by prediction/correction using the Jacobian matrix. The solution (updated moving co-ordinates, transfer function of order zero) is numerical and the inaccuracy can be demanded to be any low value.

Calculation of higher transfer functions, time derivatives and dynamics can be done after that the new position has been reached. This requires additional matrix calculations, usually generalized, that means here for all co-ordinates and form parameters together [5].

3.3 Dynamic analysis

According to the principle of Virtual Work forces can only be applied at moving points. Since mass forces will be considered as external forces according to d´Alembert, it means here that just the co-ordinates of the elements can be used to apply (mass) forces. A continuous mass needs to be transformed to an equivalent lumped mass at the nodes, using a mass matrix. This will affect (simplify) the internal force distribution in the link, but the same is true for the other mass model using a centre of gravity of the rigid body. Characteristic for the FEM concept is that a centre of gravity is not required. It is even inappropriate when flexibility of the element is to be modeled for deformed links or vibration analysis.

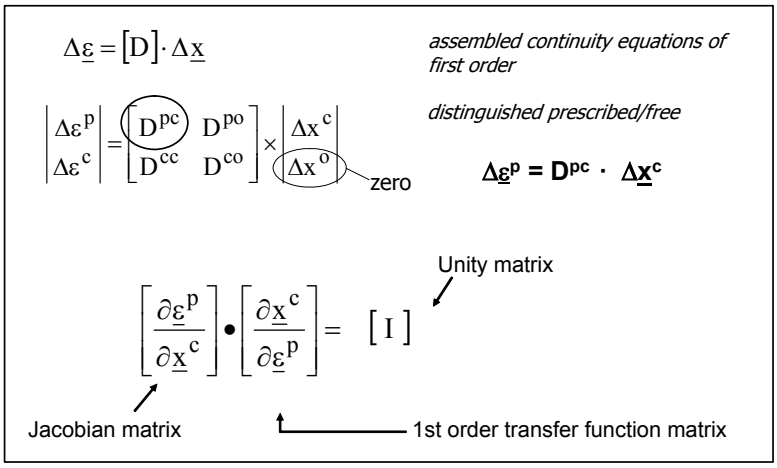

Fig. 4 First transfer function in a given mechanism position

3.4 Implementation

The current version of Runmec reads all data of the mechanism model from a data file. This is a text file, containing lines of data that has to be created by the user. With an example the implementation will be explained.

Consider the level luffing crane of fig. 5. The tip of the crane should follow a horizontal path to reduce the required power for driving the luffing motion.

Runmec can perform one of the following processes:

- Kinematic analysis
- Kinematic optimization
- Dynamic analysis
- Dynamic simulation

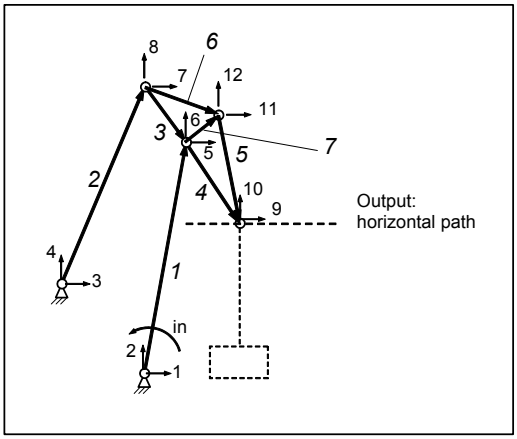

Fig.5 Runmec model of level luffing crane

	process	KIN ANA	KIN OPT	DYN ANA	DYN SIM	
	block (header name)					*required previous block*
0	VARNAMES	o	o	o	o	none
1	TOPOLOGY	+	+	+	+	none
2	GEOMETRY	+	+	+	+	TOPOLOGY
3	DYNAMICS	±	±	+	+	GEOMETRY
4	MOVEMENT	+	+/o	+	+	GEOMETRY
5	STORAGE	o	o	o	o	MOVEMENT
6	OUTPUT	o	o	o	o	STORAGE
7	OPTIMIZE	±	+	±	±	GEOMETRY
8	SETTINGS	o	o	o	o	none

Fig. 6 Runmec input file: the blocks,
Required (+), Optional (o), No effect (±)

To calculate the path of the tip „kinematic analysis“ can be selected and for this process the input file must contain the following tables (fig. 6):

- Topology (two sub-tables: element list and list of prescribed form parameters).
- Geometry (the co-ordinate values in the initial position, and optionally the value of prescribed form parameters). The moving co-ordinates may be estimated values; they will be corrected to satisfy the given values of the form parameters.
- Movement (range of steps for the input motion).

Figs 7 - 9 show details of the implementation. Each table is preceded by a block header (line with one word and no parameters). Each table line consists of a keyword and parameters. The first parameter is always numeric. When the first item on a line is numeric, the keyword is obviously missing and the previous keyword is taken. Text behind a semicolon is just comment. It has intentionally made so to help creating readable tables.

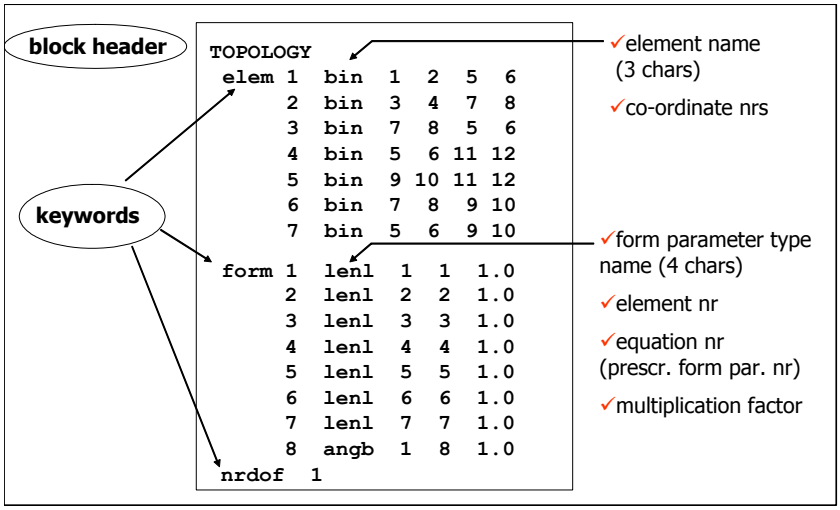

Fig.7 Specification of elements and prescribed form parameters

3.5 Data exchange

Probably many designers will prefer to create a mechanism model by graphical, interactive input. It is certainly the intention to extend Runmec in that direction in future. For data exchange of a mechanism model however a precise description of the mechanism in a text file seems to be the only workable solution. The input data file of Runmec should to be regarded also for the option to convert the information to other mechanism models. It is presented here just as an example to see how such a file could look like.

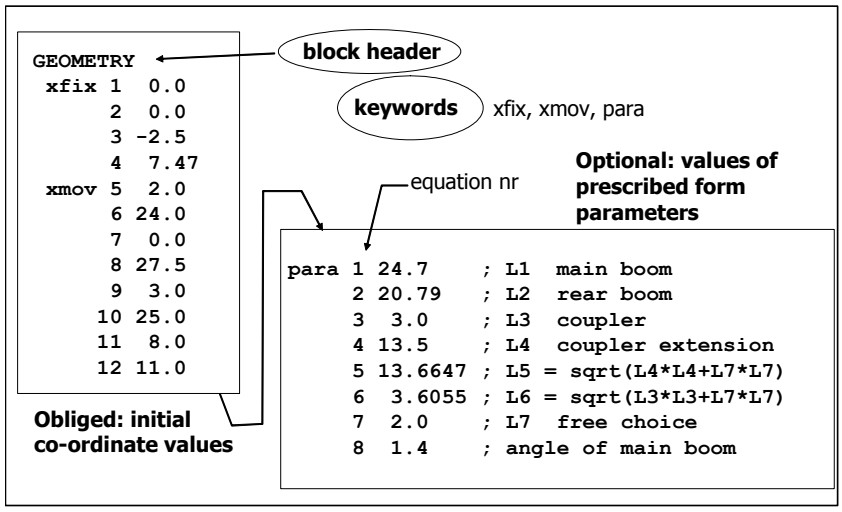

Fig.8 Specification of initial geometry and form parameters

When the conversion is to be done manually, the readability of the file will play an important role. When the conversion can be done automatically (by a computer program), the model data can also be in a less readable form like a database or an XML-file.

4 FUNCTIONS IN RUNMEC

Runmec uses functions for input and output at several places. A simple example was already given in the way input motion steps for kinematic analysis can be specified. In the block „Movement“ the keyword

- TSTEPE („Time Step for Epsilon“) has a parameter for the total motion time, while
- MOV_ABSE („Move Absolute Epsilon“) has a parameter for the total step of an input to make during that time.

With these two keywords a range of input positions is specified. For calculation of just the kinematic transfer functions these input values will do. For calculation of motion as a function of time it is assumed that the input has constant velocity. In case that the input function is some function of time, another keyword (MOV_FUNC) can be used. Now a parameter is the name of a text file that contains the input steps as a function of time. This file needs to be created by the user before starting Runmec.

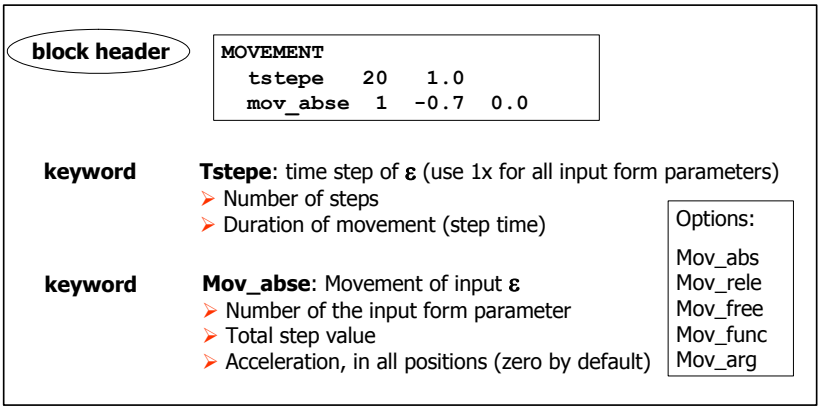

Fig. 9: Specification of input motion for batch run

Because Runmec is pure numerical (the mechanism moves from position to position) the calculated results need to be stored as far as wanted by the user. After the run the results are available for post-processing and output, to a file for instance. It would be possible to store every single array into a file, but this is not very practical. A function has often two arrays (argument and function value) or three arrays (parametric curve in 2D). Sometimes derived function values are of interest. Here starts the idea to think of a „group of arrays“ as a function object. The same need was felt when developing other computer programs by the Cadom-group at TU Delft. It finally resulted in a standardized way of maintaining functions in computer programs for mechanism design. Runmec has now various keywords, to be used in the „Storage“ block, to create such function objects for output. They can be used for input/output in Runmec and in the other computer programs of the Cadom group as well. The software to handle the basic operations on the function objects is known as the PION software, see next chapter.

5 PION FUNCTION MODEL AND SOFTWARE

PION is an acronym for Periodic functions, Input, Output and Numerics. It deals with a set of conventions for functions used in mechanism design. The idea is that a function object contains not only multiple arrays of data, but also has attributes to characterize it. Attributes concern properties like:

- A type identification (argument list, function, curve),
- Units (for argument, function, curve components)
- Periodicity (for functions) or closure (for curves),
- Derivative information,
- Name for identification of purpose.

In mechanism design many kinds of Function Objects (FOBs) can be considered. In PION subtypes have been defined to distinguish between:

- Discrete or continuous representation,
- Equidistant or non-equidistant arguments (for discrete subtypes)
- Periodic or non-periodic (for functions)
- Open or closed (for curves)
- Homogeneous co-ordinates (for infinite values) or normal co-ordinates

Examples showing the variety of these subtypes are presented in figs 10 - 13.

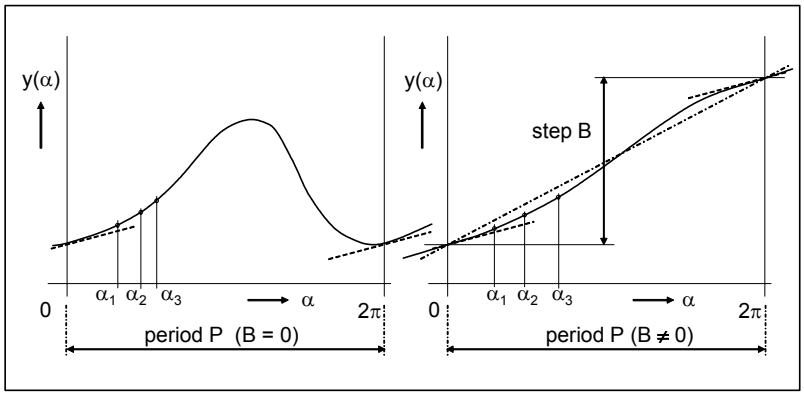

Fig. 10 Examples of periodic functions, discrete, non-equidistant points

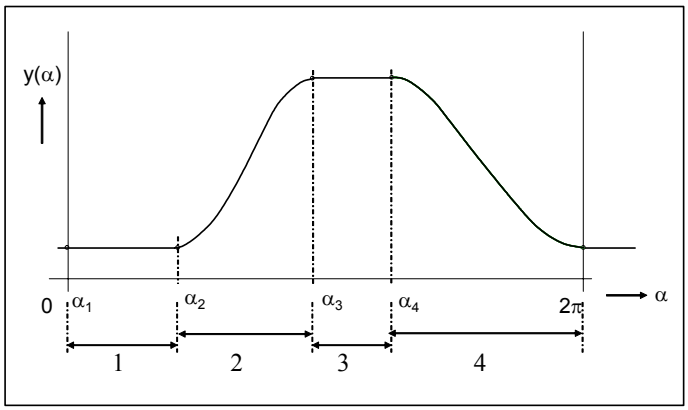

Fig. 11 Example: Continuous representation of a periodic function by a chain of polynomial functions on intervals

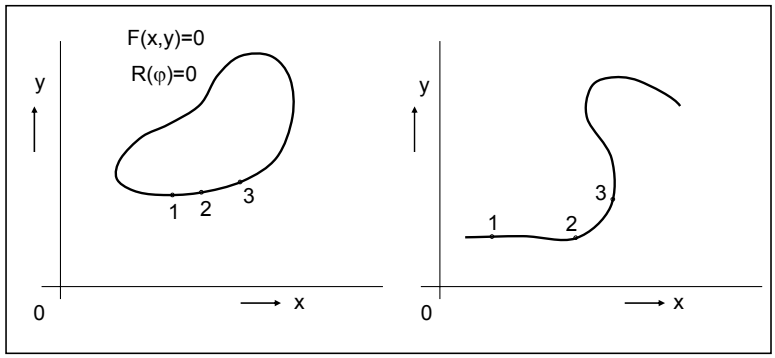

Fig. 12 Examples of Open and Closed curve, discrete

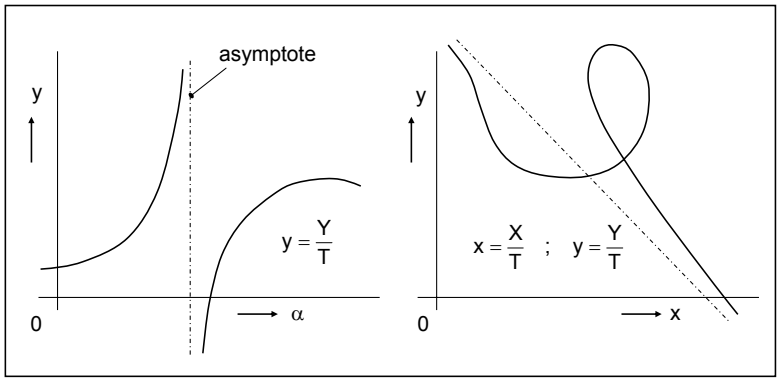

Fig. 13 Examples with Homogeneous co-ordinates (X, Y, T) for functions and curves with points in infinity

In PION the identification of all subtypes has been comprised in a 3-character code. The code F=PL2 means for instance that the Function object type is (=) a Periodic function, List of points (discrete subtype), 2nd variant (non-equidistant, normal co-ordinates).
Fig 14 shows the lay-out of function object data in a file. The most relevant syntax conventions are:

- All attribute information is placed on the first line.
- Attributes have a name (1 or 2 characters), their „value“ is added behind the „=“ character.
- Dimensions of the arrays appear on the second line.
- Following lines are for the array values. Dependent on the object subtype, one or more of the four arrays named A (argument), X (curve component), Y (function or curve argument) and T (homogeneous co-ordinate) are expected. On the same line derivative values are listed, as defined by the D= attribute for derivatives.

The PION software to handle the function objects contains basically a set of fixed data tables and a set of subroutines. The data tables concern for instance:

- Table with type codes (the 3 identifying characters) and default attribute values;
- Tables with unit codes, unit types and conversion values between units of the same kind;
- Table with derivative codes;
- Tables with predefined polynomial functions for chaining.

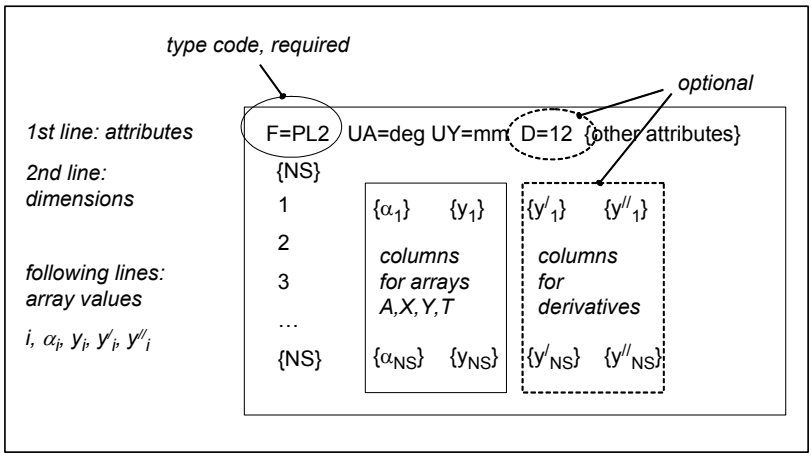

Fig. 14 Layout of a PION-file (example)

Typical tasks for which subroutines have been written are:

- Read PION object file into internal memory of the program;
- Write PION object of program to data file;
- Check consistency of attributes and data values;
- Convert FOB for other function type, units etc;
- Modify FOB for various actions (flip, shift etc);
- Graphic presentation of FOBs in a picture.

The PION software has been developed since the early beginning of the CADOM project of TU Delft [1]. Originally written in Fortran IV (1981) extensions/updates were made for the new programming standards Fortran 77 and Fortran 90. It is rather complete now, but still in development. The program Runmec uses the PION software but, as a pure numerical program, it does not take advantage of the unit attributes. Typically it will take/produce the arrays with units as dimensionless. Nevertheless, when Runmec is demanded to create a PION data file as output, the user may add (overrule) the unit information as known to him/her.

6 PRACTICAL TOOLS FOR FUNCTIONS

The PION software has originally been developed as a toolbox, to be linked to computer programs for mechanism analysis and synthesis. Its success stimulated the development of these programs greatly. On the other hand a need was felt to easily create such objects (preprocessing) and manipulate them (post-processing). With a text editor such actions can indeed be performed, but for practical use the PION toolbox could be extended with dedicated software for tasks like:

- Function editor, graphic-interactive. Not only to create and modify points of a function or curve, but also to modify the attributes and do actions like shift, flip and (unit) conversions.
- Picture editor, graphic-interactive, to combine functions in one picture. Aspects of visual properties (colour, linestyle, markers), automatic scaling and style of the axes need be regarded.

The author has recently started the development of a separate function editor for PION. A first, yet incomplete, version can be demonstrated (preliminary name FOBTOOLS).

A picture with several functions is of course useful as direct output of a mechanism program, so picture presentation should be integrated in that software and this has been done where useful. As a separate program it has competition of well known presentation software like Excel or PowerPoint, to which the array data, as present in the PION-files, can be copy/pasted easily. So a separate picture edit program has less priority.

7 CONCLUSIONS

For mechanism analysis and synthesis various theories exist. The current state of the art is that every concept uses its own way of modeling a mechanism, adapted to the specific theory. The mechanism data can in general not directly be used for another analysis or synthesis theory. The author has described in this paper how such a mechanism model for example exists for a (Finite) element based analysis in kinematics and dynamics. It is concluded that the problem of conversion between different types of mechanism models has hardly given attention in literature. It seem a good idea to investigate the option for a standard model ("super model") for which automatic conversion to all specific models should be possible. This paper intends to start the discussion for a standard model.
To stimulate the use of the various design concepts, their products like functions can be regarded for standardization too. This is mainly useful for data exchange of (intermediate) results. The author has certain experience with developing computer programs for mechanism analysis, synthesis and optimization. In this paper his basic ideas of a standard function model have been described. It is general enough to serve as a proposal for a standard function model for planar mechanisms.

8 REFERENCES

[1] KLEIN BRETELER, A.J. and H. RANKERS: Über das Programmieren von Getriebeproblemen – Vorschläge zur Strukturierung und Vereinheitlichung (in German). VDI-Berichte Nr. 321, 1979, pp. 27-32.

[2] WERFF, K. VAN DER, W. ZHANG and H.A. CRONE: A generic mechanism model for computer-aided conceptual design of machines. Design And Methodology (DTM) 1994, pp 191-204.

[3] WEN-JUN ZHANG: An integrated environment for CAD/CAM of mechanical systems. PhD Thesis TU Delft 1994.

[4] WAGENSVELD, L.C. VAN: Interactive modelling of design problems for kinematical analysis and optimization. MSc thesis TU Delft, Faculty OCP, 1993.

[5] KLEIN BRETELER, A.J.: Lecture notes on mechanisms (course wb3303) part 1 – Theory. TU Delft, Faculty 3ME, 2004.

[6] KLEIN BRETELER, A.J.: Lecture notes on Mechanisms course (wb3303) part 2 – User manual of Runmec and PION. TU Delft, Faculty 3ME, 2007. See www.ocp.tudelft.nl/tt/cadom/