PRADIS

WRITING PLUGIN-OBJECTS IN THE LANGUAGE FORTRAN

PROGRAM SET FOR THE AUTOMATION OF THE SIMULATION OF NONSTATIONARY PROCESSES IN THE MECHANICAL SYSTEMS AND THE SYSTEMS OF OTHER PHYSICAL NATURE

VERSION 4.3

Content

1. Description of technology.

1.1 purposes.

The technology of dynamic incorporation into the solver PRADIS is developed for the possibility to add the new models of elements, [PGO], [PRVP] without a change in the existing code of [reshatelja], and without its recompilation.

1.2 load of [repozitarija].

Prior to the start of calculation the solver loads the configurative file (**[repozitarij]**), in which is described the collection of the built in dynamic libraries (plugin), and component (models of elements, [PGO], [PRVP]), which in them are realized. Further for the models, [PGO] and [PRVP] are built the tables of the correspondence of numerical identifiers (from armcltg), and the structures, which describe components. The most important member in each structure of – the address of the global procedure, which realizes component (model of element, [PGO] or [PRVP]). Before the construction of the tables of the correspondence to identifiers all dynamic libraries load into the address space of the process of [reshatelja]. Further, during reading from [repozitarija] of identifier and name of the function of component, the address of the corresponding function searches for in the module of the loaded library, and it is recorded in necessary type table (model/[PGO]/[PRVP]), with its identifier.

1.3 initialization of plugin.

Besides the procedures, which realize components, each dynamic library of plugin is obligated to contain the function of initialization. In it before the calculation the solver will transmit the addresses of the terms of its common of the regions (not determined, NOTAT, GRCONF), which can be required by components in the course of computation. The function of initialization must preserve the transmitted addresses in the global variables, which will be accessible to function-components.

1.4 use of plugin with the calculation.

When resolving before the turning to the model of element, [PGO], or [PRVP] is carried out the search in the appropriate table of the loaded components on the identifier. If the corresponding to identifier component is found – it is caused it, if is not found – it is done the attempt to cause the component, turning to which is rigidly coded in the solver.

2. [Repozitarij] of plugin.

2.1 size of [repozitarija].

After the installation PRADIS of [repozitarij] of plugin it must be located in the file % OF DINSYS % \ of dinama \ of sysarm \ of plugin_repository.xml. The size of file is simplified XML. In the beginning file compulsorily must be present TEG <?xml of version= " 1.0 "?>, then one root TEG of <root></of root>. In it can be found the arbitrary collection of the subtrees XML of TEG. From them are analyzed only the subtrees with upper TEG “of plugin” (see below). The remaining subtrees of TEG can be arbitrary, size does not set on them limitations. The code of working [repozitarija] is contained in the file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of itgdll \ of plugin_repository.cpp.

2.2 tree of [repozitarija].

The principle of working [repozitarija] is the following. Entire xml file is read into the tree of the assemblies, which correspond to TEG. Tree is represented by the template of ctm::cxx::Tree, determined into pradis \ of include \ of ctmstd \ of cxx_tree.h. As the unit of tree is used the structure of ctm::pradis::plugin::Node, determined into plugin_repository.cpp. Unit stores line name, and also doubly connected list of the values of the simple types: char, unsigned of char, bool, short, unsigned of short, long, unsigned of long, int, unsigned of int, float, double, std::string. If in XML is encountered any TEG with exception of one of the special (see below), to the current unit of tree daughterly unit with the same name as met TEG, is added. Attributes and text of TEG are ignored. All daughterly XML TEG will be added to the tree as the daughterly units of already this added unit. If special TEG is encountered, then to the current unit of tree is added not daughterly unit, but sequential value (interpretation of the text of TEG), in accordance with the table:
<c></c> char,
<uc></of uc> of unsigned of char,
 bool,
<h></h> short,
<uh></of uh> of unsigned of short,
<l></l> long,
</of ul> of unsigned of long,
 int,
<ui></of ui> of unsigned of int,
<f></f> float,
<d></d> double,
<s></s> std::string.
To add value into the unit is possible pattern method of ctm::pradis::plugin::Node::push_value (). After unit is formed it is possible to sort out its values with the aid of [iteratorov] of list, of ctm returned::pradis::plugin::Node::begin () and ctm::pradis::plugin::Node::end (). To determine the type of value according to [iteratoru] is possible with the aid of ctm::pradis::plugin::Node::get_tag (), and to obtain value on [iteratoru] - with the aid of pattern method of ctm::pradis::plugin::Node::get_value ().
Thus, on the basis of [repozitarija] to be built the tree of assemblies with the names of TEG, and by the massifs of the values of the simple types, obtained from special [podtegov]. The construction of tree achieves a method of ctm::pradis::plugin::Repository::LoadTree () from plugin_repository.cpp. Class of ctm::pradis::plugin::Repository represents in solver [repozitarij] of plugin, and its public method Of load () carries out entire load of [repozitarija] (together with the construction of the tables of components), from the input flow to XML.

2.3 selection of the units of the tree “of plugin”.

After the construction of tree, the method of ctm::pradis::plugin::Repository::Load () sorts out the units of the tree of the first level after root. From them are analyzed the units with the name “of plugin”, each of which it represents one dynamically loaded library. The remaining units of the first level are ignored. In each unit “of plugin” must be located one unit “of library”, the first value in which must be the line (TEG of <s></s>) of – the name of the dynamic library of plugin, which must be located in % DINSYS % \ of dinama \ of pradis32. In addition to this, in each unit “of plugin”, can be present one subassembly “of init” even one subassembly “of cleanup”. The first values of these units must be the lines of – the names of the functions of the initialization of dynamic library (in the unit “init”) before the calculation, and its cleaning (in the unit “of cleanup”), after the end of calculation. If these units be present, functions with the appropriate names compulsorily must be exported by the dynamic library of plugin. The function of initialization compulsorily must be present in the library of plugin. If the unit “of init” there are no – it is considered that the function of initialization is called “INIT”. On the arguments of the function of initialization see below. Of functions against cleanings it can not be in the library. If there is no unit “of cleanup” - it is considered that there is no function, and it is not caused. The function of cleaning does not have arguments. Besides the units “of library”, “init” and” cleanup ", in the unit “of plugin” can be contained a arbitrary quantity of units “of model”, “pgo” or “prvp”, (describes the components, realized into plugin to library), and also any other units, which are not analyzed with the selection of the tree, built from XML of [repozitarija].
Example XML for describing plugin:

```
<plugin>
        <library><s>balka</s></of library>
        <init><s>INIT</s></of init>
        <cleanup><s>CLEAN</s></of cleanup>
        <model>
        .....
        </model>
        .....
        <pgo>
        .....
        </pgo>
        .....
        <prvp>
</plugin>
```

The code of the selection of the units of the tree “of plugin” is contained in the method of ctm::pradis::plugin::Repository::ProcessPlugin (). In this method is carried out the circuit of the subassemblies of the unit “of plugin” and the call of processors for the subassemblies of the type “model”,” pgo ", “prvp”. To if necessary add one additional type of plugin of component, this method is simple to enlarge.

2.4 selection of the units of the tree “of model”.

The code of the selection of the units of the tree “of plugin”/” of model " is contained in the method of ctm::pradis::plugin::Repository::ProcessModel (). In it the subassemblies of the following types are analyzed:

- “id”, compulsorily must be present, and contain as the first value of unsigned of int (XML TEG of <ui></of ui>) of – the identifier of model, corresponding armctlg.

- “procedure”, compulsorily must be present, and contain as the first value of std::string (XML TEG of <s></s>) – the name of the procedure of the model of element. On the parameters of the procedure of the model of element see below.
- “parameters”, compulsorily must be present. Can contain on one unit of the type “ext”, “ent”, “adr”, “ign”. The units of other types can be contained into “parameters”, but they are not analyzed. If it is present, each of the units “of ext”, “ent”, “adr” or “ign” must contain one value of int (XML TEG of). These values make the same sense as the analogous parameters of the passport of the model of element, and they must coincide with the values of the passport of this model into armctlg. If any of the values is absent, it starts that it is equal to value on silence, according to the same rules, on which are advanced the values on silence in the passport of the model of element during the addition to armctlg.
- “aliases”, compulsorily must be present and contain 0 or more than the values of std::string (XML TEG of <s></s>) – of alternative names for the procedure of the model of element, for the future use in the translator.
- “classes”, if it is present, can contain 0 or more than subassemblies of the type “system” (physical system). The units of other types can be contained into “classes”, but they are not analyzed. Each of the units “of system” can contain one unit “of name” (name of physical system) and “defaultPGO” ([PGO] on silence for this physical system). The units of other types can be contained in the units “of system”, but they are not analyzed. From the units “of name” and “defaultPGO” are extracted and are memorized the first line values, for the future use in the translator.
- “nodes”, if it is present, can contain 0 or more than the subassemblies “of node”. The units of other types can be contained into “nodes”, but they are not analyzed. In each unit of the type “node” is analyzed one unit of the type “system” (name of physical system for the unit with the number, which corresponds to the ordinal number of the unit “of node” in the unit “of nodes”). If the unit “of system” is present, the first line value is extracted from it, and it is memorized, for the future use in the translator.

Example XML for describing model:

```
<model>
	<id><ui>75</ui></of id>
	<procedure><s>MODEL</s></of procedure>
	<parameters>
		<ext><i>6</i></of ext>
		<ent><i>0</i></of ent>
		<adr><i>1</i></of adr>
		<ign><i>2</i></of ign>
	</parameters>
	<aliases>
		<s>BALKA</s>
		<s>balka</s>
	</aliases>
	<classes>
		<system>
			<name><s>mechanics</s></of name>
			<defaultPGO><s>PGO1</s></of defaultPGO>
		</system>
		<system>
			<name><s>hydraulics</s></of name>
			<defaultPGO><s>PGO2</s></of defaultPGO>
		</system>
	</classes>
```

```
<nodes>
        <node>
                <system><s>mechanics</s></of system>
        </node>
        <node>
                <system><s>hydraulics</s></of system>
        </node>
</nodes>
</model>
```

Any other units can be contained in the tree, but they are not analyzed. To if necessary add one additional characteristic of the model of element, the method of ctm::pradis::plugin::Repository::ProcessModel () is simple to enlarge.

2.5 selection of the units of the tree “of pgo”.

The code of the selection of the units of the tree “of plugin”/” of pgo " is contained in the method of ctm::pradis::plugin::Repository::ProcessPGO (). In it the subassemblies of the following types are analyzed:

- “id”, compulsorily must be present, and contain as the first value of unsigned of int (XML TEG of <ui></of ui>) of – identifier [PGO], corresponding armctlg.
- “procedure”, compulsorily must be present, and contain as the first value of std::string (XML TEG of <s></s>) – the name of procedure [PGO]. On the parameters of procedure [PGO] see below.
- “aliases”, compulsorily must be present and contain 0 or more than the values of std::string (XML TEG of <s></s>) – of alternative names for the procedure [PGO], for the future use in the translator.

Example XML for describing pgo:

```
<pgo>
        <id><ui>5</ui></of id>
        <procedure><s>AKLAB</s></of procedure>
        <aliases>
                <s>PGO1</s>
                <s>pgo01</s>
        </aliases>
</pgo>
```

Any other units can be contained in the tree, but they are not analyzed. To if necessary add one additional characteristic [PGO], the method of ctm::pradis::plugin::Repository::ProcessPGO () is simple to enlarge.

2.6 selection of the units of the tree “of prvp”.

The code of the selection of the units of the tree “of plugin”/” of prvp " is contained in the method of ctm::pradis::plugin::Repository::ProcessPRVP (). In it the subassemblies of the following types are analyzed:

- “id”, compulsorily must be present, and contain as the first value of unsigned of int (XML TEG of <ui></of ui>) of – identifier [PRVP], corresponding armctlg.
- “procedure”, compulsorily must be present, and contain as the first value of std::string (XML TEG of <s></s>) – the name of procedure [PRVP]. On the parameters of procedure [PRVP] see below.

- “aliases”, compulsorily must be present and contain 0 or more than the values of std::string (XML TEG of <s></s>) – of alternative names for the procedure [PRVP], for the future use in the translator.

Example XML for describing prvp:

```
<prvp>
        <id><ui>59</ui></of id>
        <procedure><s>X</s></of procedure>
        <aliases>
                <s>PRVP1</s>
                <s>prvp01</s>
        </aliases>
</prvp>
```

Any other units can be contained in the tree, but they are not analyzed. To if necessary add one additional characteristic [PRVP], the method of ctm::pradis::plugin::Repository::ProcessPRVP () is simple to enlarge.

2.7 addition of new properties and types of components.

By the advantage of the described in the preceding points approach with the construction on XML to the file of the tree of assemblies, and their subsequent analysis, is the comparative ease of the addition of new elements to the configuration of plugin. In fact, if it is necessary to add the new type of components, or new property to the already existing type of components, always it is possible to add into the appropriate place XML of file the new subtree of [imenovannykh] TEG, which contain the values of simple types (lines, the numbers, etc). After this, the file will be as before correctly read into the tree, simply the units of new types yet will not be analyzed. Then, should be added the code analyzing the units of new types into ctm::pradis::plugin::Repository::ProcessPlugin (), ctm::pradis::plugin::Repository::ProcessModel (), ctm::pradis::plugin::Repository::ProcessPGO () or ctm::pradis::plugin::Repository::ProcessPRVP (). Here will have to preserve values from the units, after determining for them the appropriate structures, or after enlarging those existing.

e. Changes in solver in the implementation technology.

3.1 global functions of working [repozitarija].

As it was spoken above, the class, which represents in solver [repozitarij] of plugin (ctm::pradis::plugin::Repository), is determined in the new initial file of the module of itgdll of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of itgdll \ of plugin_repository.cpp. In the same file are determined global functions for interaction [repozitariem]:
ITGDLL_INIT_REPOSITORY () of – to initialize (to load) [repozitarij],
ITGDLL_CLEAN_REPOSITORY () of – to clean [repozitarij],
ITGDLL_EXISTS_PLUGIN () of – to verify, does exist in [repozitarii] of component with the assigned type and the identifier,
ITGDLL_GET_MODEL_PARAM () of – to return the parameter of the model of element with the assigned identifier,
ITGDLL_INVOKE_MODEL () of – to cause the model of element with the assigned identifier,
ITGDLL_INVOKE_PGO () of – to cause [PGO] with the assigned identifier,
ITGDLL_INVOKE_PRVP () of – to cause [PRVP] with the assigned identifier.
All functions are decorated in the style C. during the compilation under Windows in them it is used agreement about the calls of stdcall. Because of this, function they can be caused out of FORTRAN of the code, compiled with the installations of compiler DIGITAL on silence.

3.2 connection of [repozitarija] of plugin into the solver.

Load and cleaning of [repozitarija] are added before and after calculation in the initial file of [reshatelja] of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of run \ of integr.for. Turning to the models of elements, [PGO] and [PRVP] is carried out into pradis \ of src \ of pradis32 \ of pradis \ of solver \ of itg \ of integrs.for (procedure FORMY, FORMO, FORMI) solver in the initial file. For the connection of plugin of components in this file, into all three functions is added the operator IF, which checks the presence in [repozitarii] of component with the identifier, which entered from [reshatelja] (call ITGDLL_EXISTS_PLUGIN ()). If component is found, is carried out the formation of the list of the parameters and the call of component (with the aid of ITGDLL_INVOKE_MODEL (), ITGDLL_INVOKE_PGO (), or ITGDLL_INVOKE_PRVP ()). If plugin of component is not found, as earlier is caused operator-switch GOTO, which carries out passage on the identifier to the call of component, to rigidly recorded into the code of integrs.for. Since the file of integrs.for automatically is generated with the aid of the modulus of bridge.exe, change Hollerith-coded in integrs.for they are introduced into the code of its generation in the files of gformi.for, gformo.for, gformy.for, located into pradis \ of src \ of pradis32 \ of pradis \ of solver \ of bridge \.

4 interfaces of plugin.

4.1 standardized calls of plugin.

In order to ensure the possibility of the dynamic incorporation of components into the solver, without its recompilation it was necessary to develop for all plugin of procedures the united, standardized interfaces. By interface of procedure is understood the agreement rel.un. call, and also the collection of its arguments. These agreements rigidly are written in the code of [reshatelja] before its compilation. The names of functions dynamically search for in the module of plugin of library after its load. Solver is used shch of the forms of the calls of libraries, proclaimed in in the file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of itgdll \ of plugin_repository.cpp:

- ctm::pradis::plugin::Repository::Library::FN_INIT of – the initialization of the library
- ctm::pradis::plugin::Repository::Library::FN_CLEAN of – cleaning the library
- ctm::pradis::plugin::Repository::FN_MODEL of – the call of the model of the element
- ctm::pradis::plugin::Repository::FN_PGO of – call [PGO]
- ctm::pradis::plugin::Repository::FN_PRVP of – call [PRVP]

All procedures do not return the value (type of the recovery of void). Function of ctm::pradis::plugin::Repository::Library::FN_CLEAN does not have arguments, the description of the arguments of remaining functions see in the subparagraphs below.

4.2 agreements about the calls and decoration.

During the compilation under Windows all functions, exported from plugin, must be decorated in the style C and correspond to agreement about the calls of stdcall, which makes it possible to cause and to realize them in FORTRAN the code, compiled with the aid of DIGITAL FORTRAN with the installations on silence. In C the code the arguments of these functions are described and are transferred as indicators, with the call or the realization in FORTRAN the code, arguments of functions by the usual method they are described as BY REAL * 8, INTEGER * 4 and the like should be focused attention that in the implementation plugin of library under Windows on C/Of c++ the exported functions in the style C, stdcall, will have a decoration of names of _function_name@N, where N of – the quantity of bytes on the stack, utilized for the transfer of arguments. With the registration in XML the file of [repozitarija] it follows or to indicate the same names (but not simply function_name), or to, for example, use def files for the more convenient decoration. So should be focused attention on the fact that in the implementation plugin of library on DIGITAL FORTRAN under Windows, in accordance with the recommendations in the following point the names of all exported functions will be transferred into the upper register. To avoid errors one should after assembling of plugin of library examine the decorated names of procedures with the aid of the utility of dumpbin under Windows, from [distributiva] MSVC (dumpbin/of exports of name.dll). Unix the analog of – utility nm. In XML the file of [repozitarija] the names of procedures for plugin should be prescribed in accordance with the conclusion of these utilities.

4.3 arguments of the call of initialization.

The call of initialization transfers into plugin the library of the address of the terms COMMON of the regions of [reshatelja]: not named,/NOTAT/,/GRCONF/. Thus, the list of the arguments of the function of initialization is the following:
(not named)

TIME, STEP, STEP01, STEP02, SMIN, DABSI, DRLTI, STEPMD, TIMEND, NAME, NSTEP, SYSPRN, NITER, ITR, CODE, NUMINT, NUMPP, CODSTP, CODGRF, NEWINT, MINSTP,
(/NOTAT/)
RLMAX, RLMIN, INTMAX, MSHEPS, PI, REZB, REZC, REZD,
(/GRCONF/)
RELYX, XNMPXL, YNMPXL, XNMSMB, YNMSMB, NCOLOR, NMVPAG, MODES, IK4, IS4.
Precise sense and FORTRAN the types of arguments can be looked in the description of the terms COMMON of regions from the documentation on the development of components into pradis, located into pradis \ of res \ of lsv_pradis \ of pradis \ of docs \ of include \. If plugin realizes on C/Of c++ FORTRAN the types of the arguments of the call of initialization they are mapped onto types C as follows:
REAL * 8: double *
INTEGER * 4: int *
INTEGER * 2: short *
CHARACTER * N: char *, int * (**two parameters!** They go consecutively, the second makes sense of the length of line and is equal to N).
Attention! In the implementation the function of initialization one should remember that to preserve in global variable plugin is necessary of address, but not the value of the transmitted terms COMMON of regions. In the implementation plugin on C/Of c++ this means that memorized should be the transmitted indicators. In the implementation plugin on FORTRAN is recommended the creating of the analogous TO COMMON described above regions, which contain the indicators (POINTER) of the corresponding types. Inside the call of initialization should be tied these indicators to the transmitted in the arguments of call values.

4.4 arguments of the call of the model of element.

Let us transfer the arguments of the call of the model of the element:

- The I: the vector of forces (moments) for the element.
- Y: the jacobian of the model of element.
- The X: the vector of displacements of the units of dimensionality EXT+ENT. It is not used with ADR=2, ADR=3.
- The V: the velocity vector of the units of dimensionality EXT+ENT. It is not used with ADR=3.
- A: the G-vector of the units of dimensionality EXT+ENT.
- PAR: the massif of the parameters of model.
- NEW: the vector “new state” of model.
- OLD: the vector “old state” of model.
- WRK: working massif for the model of element.

All arguments have a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++. It is in more detail about the value of arguments (with exception of the X, the V, A), and the parameters of the passport of the model of element (EXT, ENT, ADR) see documentation on the development of components in pradis, located into pradis \ of res \ of lsv_pradis \ of pradis \ of docs \ of include \.

4.5 arguments of call [PGO].

Let us transfer the arguments of call [PGO]:

- NAMEX: the name of model or [PRVP], connected with [PGO]. Massif of gaps, with the values of the parameters of passport VPS=0 and EXT=0 (fixed graphic means). Have a

type CHARACTER * 8 in the implementation of call on FORTRAN, and char *, int * (two arguments!) in the implementation on from/[S]++.

- The I: the vector of forces (moments) for the element. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- The X: the vector of displacements of the units of the model, connected with [PGO], dimensionality EXT. It is not used with the value of the parameters of passport VPS=0 and EXT=0, or with the value of the parameter UNV>0 (in this case it is used vector INNER). Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- The V: the velocity vector of the units of the model, connected with [PGO], dimensionality EXT. It is not used with the value of the parameters of passport VPS=0 and EXT=0, or with the value of the parameter UNV>0 (in this case it is used vector INNER). Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- A: the G-vector of the units of the model, connected with [PGO], dimensionality EXT. It is not used with the value of the parameters of passport VPS=0 and EXT=0, or with the value of the parameter UNV>0 (in this case it is used vector INNER). Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- INNER: the vector of the real numbers of the making sense degrees of freedom of the model of element, connected with [PGO]. It is not used with the value of the parameters of passport VPS=0 and EXT=0, or with the value of the parameter UNV=0 (in this case they are used the vector of the X, the V, A). Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- EXT: a quantity of degrees of freedom of the model of element, connected with [PGO] (length INNER). Has a type INTEGER * 4 in the implementation of call on FORTRAN, and int *, in the implementation on from/[S]++.
- PARX: the vector of the parameters of the model, connected with [PGO]. It is not used with the value of the parameters of passport VPS=0 and EXT=0. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- WRKX: the working vector of the model, connected with [PGO]. It is not used with the value of the parameters of passport VPS=0 and EXT=0. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- PAR: the vector of the parameters [PGO]. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- WRK: working vector [PGO]. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- PARLR2: the vector of the parameters of the current layer of image. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.

It is in more detail about the value of arguments (with exception of the X, the V, A), and the parameters of passport [PGO] see documentation on the addition of components to pradis, located into pradis \ of res \ of lsv_pradis \ of pradis \ of docs \ of include \.

4.6 arguments of call [PRVP].

- XOUT: the calculated output variable or the vector of the calculated output variables. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- PAR: the massif of the parameters [PRVP]. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- NEW: the vector “new state” OF [PRVP]. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- OLD: the vector “old state” OF [PRVP]. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- WRK: working massif for [PRVP]. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- A: the common vector of the real variables of [reshatelja]. Has a type REAL * 8 in the implementation of call on FORTRAN, and double *, in the implementation on from/[S]++.
- DOFADDR: the massif of addresses in the vector A, along which are located necessary [PRVP] of the value of the degrees of freedom - displacement, speed, or acceleration. If the number of unit for [PRVP] into PRADISlang is assigned as [nomer]_[uzla] of – of displacement, if it is assigned as [nomer]_[uzla]' - speed, and if it is assigned as [nomer]_[uzla] " - acceleration. Has a type INTEGER * 4 in the implementation of call on FORTRAN, and int *, in the implementation on from/[S]++.
- NDOF: the size of vector DOFADDR. Has a type INTEGER * 4 in the implementation of call on FORTRAN, and int *, in the implementation on from/[S]++.

It is in more detail about the value of arguments (with exception A, DOFADDR, NDOF), and the parameters of passport [PRVP] see documentation on the addition of components to pradis, located into pradis \ of res \ of lsv_pradis \ of pradis \ of docs \ of include \.

5 the addition of plugin to assembling of [reshatelja].

Let us transfer the steps, which must be made during the addition of plugin of component to the solver:

- To create assembling the dynamic library of plugin. Under Windows in the medium MSVC6 + OF DIGITAL Of fortran it should be use the wizard MS Of visual Of studio. If the development of plugin of library is conducted in assembling of cantilever [reshatelja], one should locate the project of plugin of library in the subdirectory of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \. In this case it should be established in tuning of the design of way to the directory of temporary files, the output directories of assembling, as is customary in assembling of [reshatelja] (see tuning the project of the test of plugin of the library of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ of balka \ of balka.dsp). In tuning of runtime FORTRAN (or) one should establish the use of runtime as multiflow dynamic library.
- To add in assembling of the dynamic library of plugin initial file, with determination and export of the call of the initialization of the library before the calculation. In more detail about the agreements of the calls of plugin of libraries, the transfer of their arguments and the size of the call of initialization see the preceding point. It is important to remember that the determination of the function of initialization must be encountered in the codes of plugin of library exactly one time, in contrast to the calls of the components (models of elements, [PGO], [PRVP]), which there can be a arbitrary quantity. If plugin library realizes under Windows on DIGITAL FORTRAN, it should be use the file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ of init.inc, in which is determined and [eksportirovana] the procedure of the initialization (see the start of file in the initial file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ of balka \ of balka.for). In the file of init.inc, transferred [reshatelem] the members COMMON of regions tie to FORTRAN to the indicators, which are placed into analogous COMMON of region. If plugin library realizes on C/Of c++, it should be use the file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ of init.h, in which is determined and [eksportirovana] the procedure of the initialization (see the start of file in the initial file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ md \ of main.cpp). In the file of init.h, transferred [reshatelem] the indicators to the terms COMMON of regions remain in the global variable of – aggregate from these indicators.
- To add in assembling of the dynamic library of plugin initial file, with determination and export of calls of one, or several plugin of components (models of elements, [PGO], [PRVP]). In more detail about the agreements of the calls of plugin of libraries, the transfer of their arguments and the sizes of the calls of components see the preceding point. If plugin library realizes under Windows on DIGITAL FORTRAN, it should be use the file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ of common.inc, in which are determined BY COMMON of region from the indicators FORTRAN, corresponding COMMON to the regions of [reshatelja]. The values of the terms of these COMMON of regions can be required plugin to components with the calculation. See the start of the file in the initial files of plugin of the component of balka.for (model), aklab.for ([PGO]), X.for ([PRVP]) realizing test. In the files it is possible to borrow the method of the export of the calls of components with the aid of the special commentary (for example!DEC$ ATTRIBUTES DLLEXPORT::AKLAB). If plugin library realizes under Windows on C/Of c++, it should be use the file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ of common.h, in which is determined structure- aggregate from the indicators to variable COMMON of regions [reshatelja]. The values of the terms COMMON of regions can be required plugin to components with the calculation. For the access to the indicators to the terms COMMON of regions it follows to use macro PLUGIN_COMMON as by the name of

the variable- structure, terms of which are indicators. See a example of the start of the file of common.h, and the application of a macro PLUGIN_COMMON in the initial file of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ md \ of main.cpp, which realizes test plugin to component (model of element MD). In the project of pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ md also it is possible to borrow the method of exportation and decoration of the call of component in the dynamic library with the aid of .DEF of file.

- To register component in the binary catalog PRADIS of armctlg. For this it follows to use the utility of arm with the parameter!. It should be caused in the directory, where is located file with the expansion .FOR, which is begun from the special commentary of component. If plugin of component realizes to C++, for the registration with the aid of arm should be created the file with the expansion .FOR, which contains only this special commentary. After registration with the aid of arm, should be determined armctlg the identifier of component, for the subsequent introduction in [repozitarij] of plugin (see the following subparagraph). For this he is proposed to open the modified file of armctlg by the editor of the binary files MS Of visual Of studio. In the file should be found the first entry of the name of the added component and memorized the number of the line of entry (to the left in 16- [richnoj] form). This number one should transfer into the decimal form, divide by 80, and add 1, for which, for example, it is possible to use standard calculator Windows. So will be obtained armctlg the identifier of component.
- To register in [repozitarii] of plugin library, call of its initialization and calls of plugin of components realized in it. See the point, which describes of [repozitarij] of plugin, and also the example to the registration of test plugin of libraries. On obtaining of the identifier of component for the registration see in the previous subparagraph.

If into plugin components is required the call of the library functions of [reshatelja] (for example, S00X), they cannot be caused from the static libraries, directly connected by c plugin by library. This because then the library functions can attempt to use COMMON by the regions of [reshatelja] directly, and they in the module of the dynamic of plugin of library be absent. In order to use library functions in the realizations of plugin of components, it is necessary to carry out their “wrappers” into the module of itgdll and to connect plugin library during the assembling from itgdll.lib. Then with the call of the function of plugin library will be turned to the module of itgdll, and there narrower than function they will be correctly caused. See a example of call from pradis \ of src \ of pradis32 \ of pradis \ of solver \ of plugin \ of balka \ of aklab.for of functions DRAWAB and W_GLASS, which are the wrappers of real library functions and are determined into pradis \ of src \ of pradis32 \ of pradis \ of solver \ of itgdll. At the present moment in itgdll is included the limited set of the library functions, which were required in the implementation test plugin of the libraries of balka and md. It is assumed that subsequently all necessary functions will be wrapped up by the analogs, determined into pradis \ of src \ of pradis32 \ of pradis \ of solver \ of itgdll \ of libdllexp.f90.