# Using LEDAS Computational Software Toolkits to Shorten Development Cycle of Variational CAD Systems

Dmitry Ushakov

## From Constraint Solver to Software Toolkits

In the paper we present the strategy of the LEDAS Company aimed at providing ready-to-integrate toolkits of computational software modules for developers of commercial CAD/CAM/CAE systems. The modules provided by LEDAS are supposed to be integrated into an application for implementation of the so-called variational approach to parametric design (further on referred to as variational design). The paper describes the main advantages of variational design over traditional history-based parametric design. It presents the main computational engine a geometric constraint solver and discusses its applications in CAD/CAM/CAE domain. Describing the basic functionality of typical CAD/CAM/CAE applications (2D sketcher, 3D part modeler, assembly design, drafting, kinematics analysis, and others), we emphasize the advantages of variational approach for end-users. In conclusion we list some prospective applications of variational design in other domains.

## Content

ledas

24th August 2007

Using LEDAS Computational Software Toolkits
to Shorten Development Cycle of Variational CAD Systems

## Parametric Design: Variational Approach

Parametric design is a revolutionary paradigm of CAD systems (including mechanical, electrical and architectural CAD), which allows users to change the geometric shape of a designed product by assigning new values to its parameters. The common approach to modeling 2D and 3D shapes is to use a set of standard features. A feature defines a particular local property of a product. The simplest features are single geometric elements such as points, segments, arcs, or patches. More complex features correspond to operations of creation and editing 3D shapes like sweeping a 2D profile along a 3D curve, faceting a sharp edge, boring a hole, etc. Such complex features depend on other ones. All features of a geometric model form so called feature tree representing the dependencies between them. Each feature has a predefined set of parameters (point coordinate, fillet radius) which values can be changed by a user. In that case the so-called update mechanism is used to regenerate a 3D shape. The feature which parameters were changed is rebuilt (by regenerating a corresponding geometric shape from new values of parameters) as well as all features depending on it. E.g., when a user changes a 2D profile, a pad (a prism) based on this profile is also rebuilt as well as all fillets made on its edges. Traditional feature-based parametric design is called history-based, since its update mechanism is based on a design history. This design paradigm is easy to implement, although not so easy to use: when users need to position two features w.r.t. each other, there is only one way to do that: at the time of their creation. Cyclic dependencies between features are not allowed in the history-based design paradigm, since they lead to infinite looping of update mechanism. However, cyclic dependencies usually constitute a natural way to express the design intent. A history-based model is always fully defined: there is no room for freedom of its parts (features). Users have only one possibility to edit the geometric shape by changing the features values.

In contrast to traditional history-based geometric modeling, a new parametric design paradigm is proposed the so-called variational approach. With this approach, a user puts arbitrary geometric constraints between features and their parameters. A constraint is a logical or parametric relation specifying possible relative positions of its arguments. Logical constraints include coincidence, parallelism, and tangency between geometric entities. Parametric constraints include measures like distance, angle, and radius. Constraints play central role in the variational approach: the CAD system should automatically change geometric parameters (which still control the geometric shape) to satisfy all user constraints. Specifying constraints is easier than manually changing the values of parameters: to position two elements (features) w.r.t. each other a user should only put the corresponding constraint(s) between them. The design history does not matter at all, cyclic dependencies are naturally allowed, and remaining degrees of freedom can be used to interactively modify the geometric shape by dragging particular feature(s) with a computer mouse. However, the most important advantage of the variational design is possibility of modifying a model by a user who did not design it, since there is no need to understand the design history and design intent. The key differences between two approaches to parametric design are listed in Table 1.

| **Parametric modeling based on…** | |
|---|---|
| **…design history** | **…variational approach** |
| The way a feature is positioned w.r.t. other ones is defined at the moment of its creation | Relative positioning of several features can be changed at any moment by means of constraints |
| The only way to change the position of a feature is to edit the values of its parameters | Feature parameters can also be changed manually; usually they are changed automatically as a reaction on imposed constraints |
| The result of editing a feature does not change the position of the ones created before | All features are equal: editing a particular feature can involve updating all features (independently on the order of their creation) |
| Cyclic dependencies between features are not allowed | Arbitrary cyclic dependencies between features are allowed |
| A geometric model is fully defined only one feature per time can be edited | The remaining degrees of freedom can be used for dynamic dragging of features while keeping all imposed constraints satisfied |
| To update a model after the change of a parameter, the CAD system performs only one traversal of the feature tree | To update a model after imposing new constraints or changing the values of the existing ones, the CAD system solves a system of simultaneous equations |

Table 1. Two approaches to parametric design

Certainly, the main drawback of the variational approach should be mentioned the need to solve efficiently large systems of simultaneous constraints (represented algebraically as non-linear equations over features parameters). Traditional numerical methods for solving the systems of non-linear equations cannot guarantee the desired scalability (solving time should be in direct proportion to the number of constraints), naturalness (the proposed solution should correspond to the user's pragmatic expectations), and solvability (in case of no solution for a set of constraints, a partial solution should be given, which satisfies as many constraints as possible). Mathematical methods, which guarantee the desired properties, has been intensively scrutinised by the leading research groups in the field of geometric constraint solving since 1960s. Many techniques have been developed, especially those aimed at decomposition of an initial large constraint satisfaction problem into a set of smaller ones: solving them one by one using optimal algebraic modeling, it is possible to find an efficient solution to the whole problem. LEDAS know-how in this field is based on both the experimental works conducted at the Russian Research Institute of Artificial Intelligence and research performed at LEDAS during the last 8 years and resulted in development of a geometric constraint solver called LGS, which serves as the base for efficient implementation of the variational approach in several commercial CAD systems.

## Geometric Constraint Solver

A geometric constraint solver is a software module which takes a set of geometric objects and constraints as an input and computes new positions for each object which satisfy all (or most) imposed constraints as an output. LEDAS geometric solver, LGS, implements a set of efficient methods to solve large industrial instances of constraint satisfaction problems. Using several heuristic techniques for problem decomposition, simplification and algebraic modeling, it solves instances involving hundred constraints interactively and thousand constraints in few seconds. Its application programmer's interface (API) allows integrating LGS into any CAD application for implementing the variational approach to parametric design.

Basic LGS entities include planimetric and stereometric geometric objects points, (infinite) lines and planes, circles, cylinders, spheres, cones, etc. Each object is defined by its geometric parameters (Cartesian coordinates, radii, etc.) Arbitrary application-defined curves and surfaces are allowed, including the discontinuous ones and NURBS. Such curves and surfaces are defined by providing (on the application side) a function that evaluates a point on a curve/surface for the given local parameter(s), varying in the given limit(s). The absolute or relative position of some geometric objects in 2D/3D space can be fixed, a rich set of constraints between them is allowed: coincidence, tangency, parallelism, perpendicularity, concentricity, or coaxiality. In addition to the logical constraints, one can define any desired values for constrained dimensions (distances, radii, and angles).

Apart from pure geometric entities LGS supports free parameters and free dimensions. Their values are not defined explicitly, but computed from the so-called engineering constraints imposed on them: equations and inequalities involving standard or application-defined functions, and design tables (where possible tuples of values are listed).
Basic functionality of the LGS solver consists of finding such values for all geometric and engineering parameters, which satisfy all imposed constraints (or most of them). In case of several possible solutions, LGS provides the most natural one: new coordinates/parameters should be as close as possible to the initial ones, all initial relative orientations of objects (the so-called chirality) should be kept.

An important additional functionality is over-defined and inconsistent constraint diagnostics, which allows a CAD application user to see and fix all errors in constraint-based specifications of a designed product.

Move under constraints functionality consists of finding a movement trajectory of the given object(s), which is as close as possible to a user-defined movement. All imposed constraints should be satisfied at each intermediate point of the trajectory. Such functionality is important for several typical CAD applications, including sketching, assembly design, and kinematics simulation.

LGS is a cross-platform software. It consists of a set of binary libraries, which implement the above listed objects and functionality. The LSG API (Application Programmer's Interface) is a set of functions with pure C interface, so the solver can be integrated into an application coded in C, C++, C#, Java, VB, and many other programming languages. Binary versions of libraries are available for Windows, Linux, *BSD, and AIX platforms. LGS can be easily ported into any other platform, since its code uses only industrial subset of the C++ programming language, available with most C++ compilers.

LEDAS plans to develop several additional software modules, which extend the basic functionality of LGS (some of them have been already implemented): kinematic joints modeler, 2D profile manager, Boundary representation (BRep) interface for LGS, geometric measurer (which computes lengths, areas, volumes, inertia tensors, etc.), collision detector, and the like. In a single package with LGS itself, these modules form the full portfolio of computational software components needed to develop most variational CAD applications from ready-to-integrate set of toolkits.

## Development of Variational Design Applications

Despite a lot of geometric software components available in the market, development of new CAD applications still remains a resource-intensive task, which includes integration of modules from different vendors, development of own modules, testing and debugging of the whole application in cooperation with maintenance service departments of other companies. To cut down overall expenses on CAD development and maintenance, LEDAS experts have analyzed customers' experience in building LGS-based applications and defined a set of guidelines. Soon LEDAS will propose ready-to-integrate toolkits (sets of software modules) together with detailed explanation of how to use them for developing typical

variational design applications like sketcher, drafting, 3D modeling, assembly design, and the like.

A typical CAD system is always built on the base of a geometric modeling kernel. Such a kernel usually provides a set of data structures and functions to model geometry and topology for boundary representation (BRep) of solid models, implements the basic operations on creation and modification of 2D and 3D geometric shapes, includes algorithms for tessellation, hidden line removal, rendering, and supports open/save file operations for popular CAD data formats. There are several geometric modeling kernels available in the market, and LGS geometric constraint solver can be integrated with any of these kernels (as LGS itself is not a geometric modeler). LEDAS has developed a dedicated integration technology which allows using geometric modeling kernel's data structure in LGS and related software modules. The links between internal software modules of a typical integrated CAD application are shown on Fig. 1.

This way, a CAD developer has the full set of tools needed to build a variational design application based on geometric constraint solver and related software modules. LEDAS decided to group the existing and future (planned for development) modules in toolkits corresponding to typical CAD applications (some of them are listed in Table 2). Each of these toolkits includes a dedicated version of LGS constraint solver, additional computational software modules, the technology of their integration with any geometric modeling kernel, detailed programmer's guide, and an open-source sample application which demonstrates the corresponding functionality and can be used as a base for a future customer application.

Market availability of each of these toolkits will be announced later. In the remaining part of the paper we present a detailed description of each of these toolkits and explain the advantages of applying the variational approach in the corresponding CAD areas.

## Sketcher

2D Sketcher is used to design open and closed 2D profiles. It is a must-have application for any modern CAD system, since a 2D profile is a base feature for swept surfaces, pads, pockets, holes, and many other 3D features. A 2D profiles consists of linear and curvilinear segments. Each such segment is defined by its parameters. Usually the set of parameters is redundant, although any of them can set by user: for a linear segment these parameters are coordinates of its end points, its length and angle with horizontal direction, for a circular arc one can define its end and central points, radius and angle. In turn, the point coordinates can be defined in the Cartesian or in the polar system. A typical Sketcher application supports a rich set of 2D features: points, lines, segments, rectangles, polygons, circles, ellipses, parabolas, hyperbolas, arcs, splines, offset and connect curves.

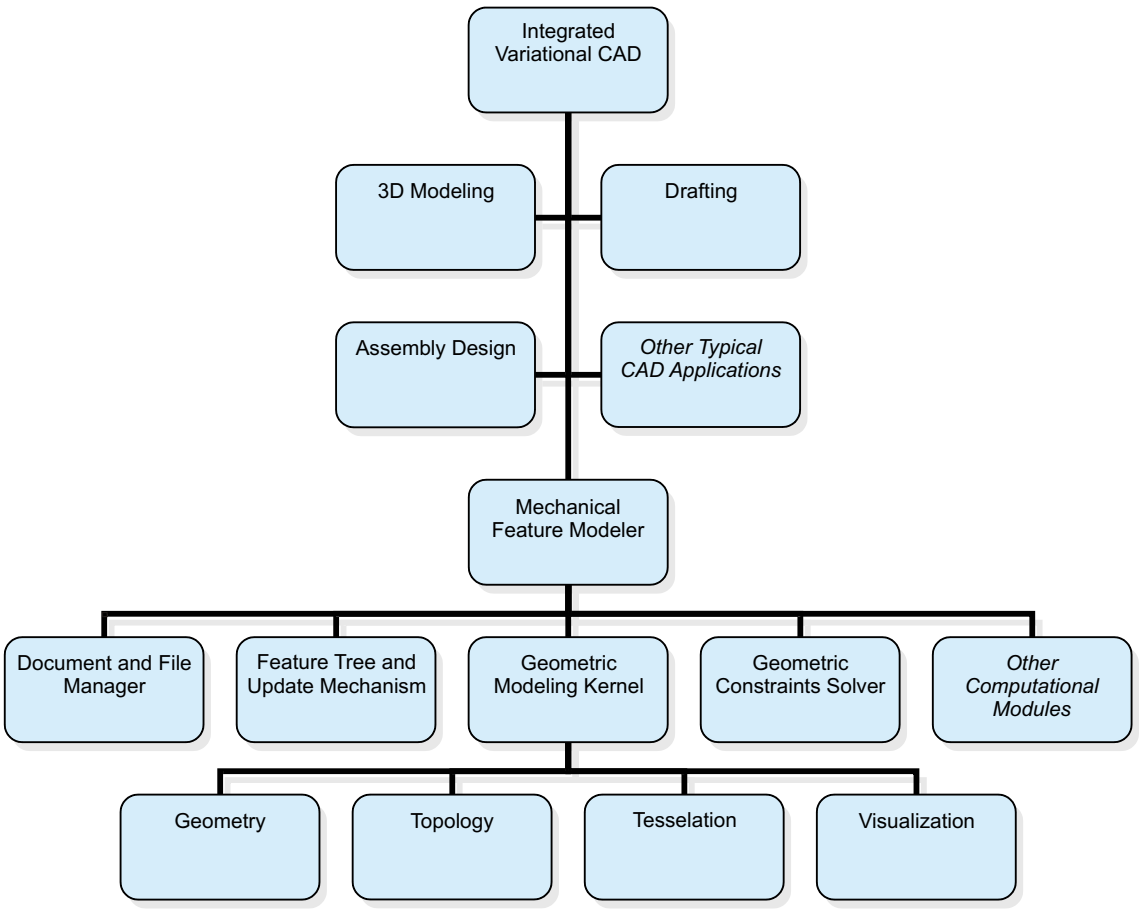


Figure 1. Software modules architecture of a typical integrated variational CAD system

KUsers of history-based parametric sketcher can control the shape of a 2D profile by changing the parameters of its elements. Variational parametric sketcher additionally allows one to control the shape with the help of logical and parametric geometric constraints. E.g., one can define two segments to be parallel, or perpendicular, or on a particular angle, or on a particular distance from each other, or tangent to a circle, etc. An arbitrary number of constraints can be imposed on all 2D elements, and the Sketcher application will automatically (usually in an interactive mode with an automatic shape update after each user's manipulation) satisfy all (or most) imposed constraints by changing the parameters of the elements to position them in the given relationship w.r.t. each other. Constraints can form the so-called cyclic dependencies (when there is a chain of elements, which next element is connected by a constraint with the previous one, and the first one is connected with the last). A typical example of a cyclic dependency is a tree points with three distance constraints between them. A typical Sketcher application is able to detect redundant (useless) and inconsistent constraints and highlight them to a user. An important advantage of the variational approach is the ability to use the remaining degrees of freedom of all 2D elements to drag them by pointing with a computer mouse: during the dragging process the application will automatically reposition all elements to follow the trajectory of the user's movement and satisfy all imposed constraints at each intermediate point of the trajectory. Such functionality is an easy-to-use and powerful way to change the geometric shape. Another, perhaps more typical possibility of variational sketchers is the ability to put the constraints automatically to remove all remaining degrees of freedom in 2D profile (the so-called auto-constraining). It is useful, when a user wants to have full control of the shape.

| **Toolkit** / **Functionality** | **2D Sketcher** | **3D Part** | **3D Assembly** | **2D Drawing** | **2D/3D Engineering** | **2D/3D Optimization** | **3D Kinematics** | **3D Dynamics** | **3D Interoperability** | **3D Configuration** | **3D Viewing** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Planimetric objects and constraints | + | + | | + | + | + | | | | | |
| Stereometric objects and constraints | | + | + | | + | + | + | + | + | + | + |
| Engineering parameters and constraints | | | | | + | + | | + | | + | |
| Kinematics joints | | | | | | | + | + | | | + |
| Constraint satisfaction | + | + | + | + | + | | | | + | + | |
| Parametric optimization | | | | | | + | | | | | |
| Diagnostics | + | + | + | | | | | | + | | |
| Degrees of freedom | | | + | | | | + | | | | |
| Move under constraints | + | + | + | | + | | + | + | + | | + |
| Auto-constraining | + | | | | + | + | | | | | |
| 2D profiles and offsets | + | | | | | | | | + | | |
| Boundary representation | | | + | | + | + | + | + | + | + | + |
| Measures (length, area, volume, center of gravity, inertia tensor) | | | + | | + | + | | + | | | |
| Collision detection | | | + | | | | + | + | | | + |
| Collision resolution | | | | | | | | + | | | |

Table 2. Configuration of several typical LGS-based toolkits for variational design applications

The 2D version of geometric constraint solver LGS models all geometric entities of a typical sketcher application in terms of planimetric objects  points, infinite lines, circles, ellipses, black-box curves (includes the non-rigid ones). With this modeling, a sketcher entity, like a linear segment is represented as an infinite line and two points, which are coincident to the line. In the same way a circular arc is modeled as a circle with two coincident points. To simplify the translation between the native sketcher representation (2D profiles) and LGS 2D entities, LEDAS provides a special source code templates (for C++ programming language), contained 90% of the customer's source code needed to integrate LGS into an application based on any geometric modeling kernel. The basic functionality of LGS 2D version for Sketcher application consists of static and dynamic (when dragging) constraint satisfaction (with computing new coordinates for all objects), diagnostics of objects and constraints, and auto-constraining. Another module of LGS 2D Sketcher Toolkit is now under development. It will model a complex 2D profile as a single entity: one can put a geometric constraint on the whole profile (not only on its particular segment or arc), or create an offset curve for a particular profile and put constraints on it, and the like. The already existing sample application called Lege'n'd 2D demonstrates the basic ideas of variational sketching: one can create parametric sketches with segments, circular and elliptic arcs, parabolas, hyperbolas and spline curves, and put constraints on them (see Fig. 1). This open-source Windows application is based on the open-source geometric modeling kernel Open CASCADE, and LEDAS customers are able to use its code for developing own sketcher-like applications, including many technical aspects related to a graphical user interface. All these modules together with LGS 2D Manual (of 100 pages long) form the so-called LGS 2D Sketcher Toolkit  a comprehensive set of software tools for developers of Sketcher applications with variational functionality.

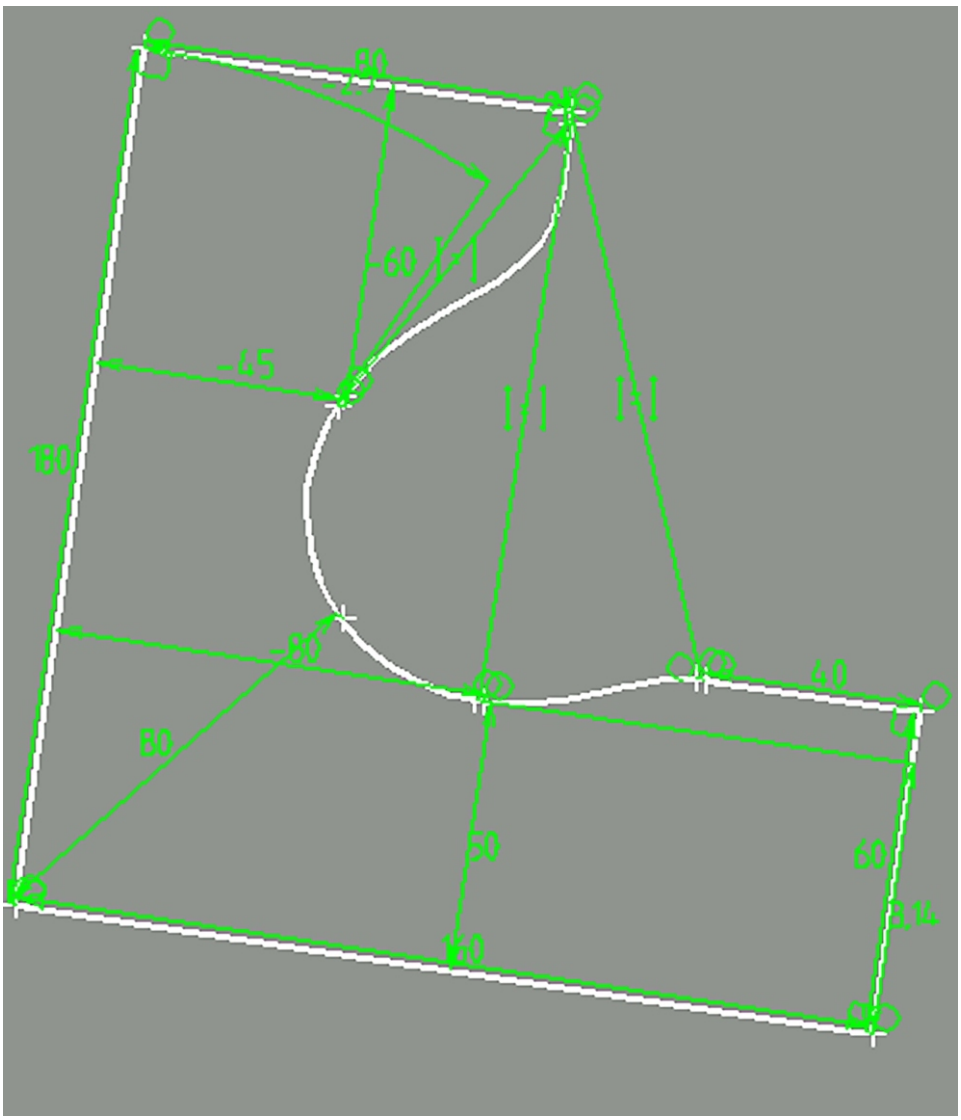


Figure 2. Using the variational approach to design a curvilinear 2D profile with a Sketcher-like application

## 3D Modeling

Variety of modern 3D modeling tools is divided into three groups: wireframe, surface, and solid modeling. The common approach here is feature-based parametric modeling based on design history. Until now, the variational approach has had a very limited usage in this area for two reasons. First, 3D modeling is much more complex than the 2D one: rich set of 3D geometric primitives (wireframe, surface, and solid features) requires more sophisticated development to model them inside a geometric constraint solver. Second, feature-based design is de facto industry standard of modern MCAD software, and traditionally (as the first implementation in a commercial system) it was history-based, not a variational one.

Wireframe design is very similar to sketching with the only difference: designed profiles are 3D, not only 2D. Therefore, the variational approach has here the same advantages over history-based design: the user must not be concerned about the design history, any constraint between any elements can be imposed at any design stage, cyclic dependencies are naturally allowed, under-constrained parts of a model can be used to drag profile elements.

Surfaces are usually designed with three main ways. The simplest surface is a patch of one of canonical surfaces (plane, cylinder, sphere, cone, torus), which is defined by the Cartesian or polar coordinates of the surface and limits for UV parameters of the patch. The second way is creating a surface on the base of one or several 2D and 3D curves. A 3D curve can be extruded along a given direction (or revolved around a given axis) to form an extruded (revolution) surface. The same way, swept surfaces are created by sweeping out a 2D profile along a 3D curve. The profile can be changed continuously during the sweeping, producing the so-called adaptive swept surface. Similarly, a fill surface is created by internal approximation of several 3D curves that form its boundaries. Finally, the third way to create a surface is to build it on the base of another surface. For example, an offset surface lies on the given distance from another surface (if the distance is changed along the surface, it is called a variable offset surface), while blended surface connects (with desired level of continuity) two other surfaces. The variational approach can be used here for several things: first, for creation of complex surfaces on the base of continuously changed elements (like adaptive swept surfaces) to control the form of the changing element; second, to position surfaces and their defining elements w.r.t. each other in 3D space. A user is able to put distance, angle, coincidence, or tangency constraints between any of those elements.

In solid modeling, commonly used features define a new solid (which is added to or subtracted from the existing one) by extruding or revolving a closed 2D profile: pads, shafts, ribs, pockets, grooves, slots, holes, and so on. Solid modeling features of the second group are based on modification of topological elements of existing solid and include fillets, chamfers, drafts, and so on. The third group form surface-based features: splits, thicks, closes. Finally, solids are modified by transformations (translation, rotation, symmetry, mirrors, pattern-based) and Boolean operations (intersection, union, difference). With the variational approach a user is able to position features w.r.t. each other in 3D space, e.g., to define an angle between axes of two holes, or 3D distance between centers of their sketching circles.

As mentioned above, many traditional feature-based modeling applications do not allow users to put constraints between geometric elements. A user is able to measure the distance or angle between two particular points, curves or surfaces, but is not able to change its value: many features are built on the basis of others (unlike Sketcher application, where all features are independent), and to resolve the constraints a geometric constraint solver should be deeply integrated with a geometric modeling kernel, since many geometric elements are the results of complex modeling operations. However, there is a simple way to overcome these difficulties and to use the variational approach in 3D modeling even without integration of constraint solving and geometric modeling to forget the design history. Indeed, if a user specifies that a particular feature has no history tree, it can be linked by 3D constraints (logical and parametric ones) with other features, since the application can modify it with no relation to other features. The corresponding features (without design history) are called the datum ones. For example, a particular datum solid can be treated as a rigid set of its topological elements: vertices (points), edges (linear segments, arcs, curves), faces (planes and other surfaces), and all these elements can be linked by constraints with other geometries in a model. This is especially useful for wireframe and surface design.

LGS 3D Modeling Toolkit aimed at simplifying the development of 3D modeling applications will consist of a 3D version of LGS geometric solver (supporting all basic 3D entities: points, lines, planes, canonical and application-defined curves and surfaces) and dedicated integration technology allowing to put constraints directly on the topological elements of the boundary representation of a wireframe/surface/solid body (with no need to translate them to LGS 3D objects).

## Assembly Design

Assembly design is an application, where the variational approach is the only reasonable choice. Indeed, the so-called bottom-up approach allows users to assemble a mechanism from already designed parts; and the most natural way to position parts w.r.t. each other is to use constraints. Four kinds of assembly constraints are commonly used: coincidence, contact, offset and angle. In addition, a user can fix some parts absolutely or relatively (by grouping some subset of parts in the so-called rigid sets). This approach to assembly design requires powerful variational functionality: arbitrary order of constraints (design freedom), cyclic dependencies (a typical mechanism has many points of contacts between parts), under-defined assemblies (moving parts always have remaining degrees of freedom), so it is the most comprehensive application for the geometric constraint solver.

Basic functionality of a typical assembly design application is automatic transformation of non-fixed parts in 3D space to satisfy all (or most) imposed assembly constraints (see Fig. 3). Designing an assembly, a user can call for update several times when adding new constraints or changing the values of their parameters.

In the framework of the variational approach, a user is able to impose many constraints between parts and obtain an over-defined or inconsistent assembly. A proper application highlights redundant and inconsistent constraints, allowing users to remove or modify them.

Typical functionality of assembly design applications is the possibility to disassemble automatically an assembled mechanism, to show the user parts it consists of.

Another typical functionality is the so-called clash detection an automatic check of intersection (with the given tolerance) of volumes of solids that form each part. Similar functionality consists of computing the minimal distance in 3D space between two particular parts and its visualization.

Each part has from 0 to 6 degrees of freedom (depending on its fixation or constraints linking it the with fixed parts). Degrees of freedom analysis and visualization is an important functionality of assembly design applications.

Parts with remaining degrees of freedom can be moved. It is important for designer to see the trajectories of such movements in accordance with assembly constraints. Usually a user specifies the desired movement (translation along a particular direction or rotation around a particular axis) for a part, and the application automatically moves the part trying to keep it as close as possible to a specified movement, in the same time satisfying all assembly constraints in each intermediate position. This functionality allows users to simulate the kinematics of the designed mechanisms.

To position 3D annotations and sections of the designed assemblies one can use 3D constraints; however, in most commercial assembly design applications the corresponding features are history-based.

Finally, many applications are able to compute different geometric and mechanical measures of parts and mechanisms: volume, mass, surface area, center of gravity, inertia tensor; some of them are based on the Bill of Materials (BOM) specifications.

In the framework of planned for development LGS 3D Assembly Toolkit, LEDAS is going to provide computational modules covering most of the functionality listed above. Already existing LGS 3D, being integrated in a commercial CAD application, efficiently solves large assemblies with thousands constraints. It supports constraints between stereometric elements (points, infinite lines and planes, canonical and application-defined curves and surfaces) grouped into rigid sets. The existing functionality includes constraint satisfaction, constraint diagnostics and move under constraints. Additional modules for computing degrees of freedom, detection of clash, computing geometric measures will be available soon.

## Drafting

Drafting in modern 3D CAD systems can be performed in two different ways: automatic draft generation from 3D models of parts and assemblies, and 2D drafting from the scratch. These two methods are often used together: basic draft elements are generated automatically; others (the so-called interactive elements) can be drawn by hand using specific drafting capabilities of a corresponding CAD application.

An important feature of drafting applications based on variational approach is setting logical and dimensional geometric constraints between 2D geometric elements of a draft. In most cases, constraints are used to position properly interactive elements w.r.t. the ones generated from a 3D model. The corresponding variational functionality of a Drafting application is similar to the one implemented in Sketcher applications (described above). However, there is another important feature of advanced drafting applications consisting in setting the so-called driving dimensions. Such a dimension (a radius, a distance, an angle) can be set between 2D elements generated from a 3D model. Changing the value of a driving dimension parameter forces re-building the 3D model by an integrated CAD application to satisfy the given dimensional constraint on a given 2D projection. Usually it can be implemented by association of a driving dimension of a Drafting application with the corresponding 2D or 3D dimension of Sketcher or 3D Modeling applications.

LGS 2D Drafting Toolkit (planned for development) is aimed to implement the link between 2D and 3D constraints as well as to support typical geometric entities of a drafting application 2D profiles, geometric dimensioning and tolerancing specifications.

## Knowledge-Based Engineering

Advanced parametric design systems have the ability to link geometric and engineering parameters with so-called engineering constraints: formulas, rules, equations and inequalities, design tables, etc. Usually these engineering specifications are built on top of parametric geometric model. It means that after resolving engineering constraints, new values of geometric parameters are used to update the geometric model (with history-based or variational approach). To simplify the resolution of engineering constraints, cyclic dependencies between engineering and geometric parameters are prohibited in many CAD systems. However, full-scale implementation of variational approach can remove such artificial limitation to ensure the engineering design freedom.

The LGS constraint solver efficiently handles not only pure geometric constraint satisfaction problem, but also mixed ones, composed of both kinds of constraints geometric and engineering. In this framework, a user can create both free parameters and parameters linked with particular geometric dimensions (radius, distance, or angle) and put engineering constraints on them equations, inequalities and tabular relations. Equations and inequalities can refer to other parameters and involve both standard math functions (so-called white boxes, since their analytical representation is known) and the application-defined ones (called black boxes, since their analytical representation is unknown to the constraint solver that evaluates them as callback functions). Inside LGS, both geometric and engineering constraints are translated into a system of equations, which can be efficiently solved, thus allowing simultaneous resolution of the whole variational model.

It is planned to implement additional feature around LGS the ability to compute so-called geometric measures: length of a curvilinear profile, closed boundary area, 3D surface area, volume of a solid, center of gravity, etc. Engineering parameters linked with the corresponding geometric measures can then be used together with free and dimensional parameters to specify different engineering constraints. Due to the ability of LGS to efficiently solve mixed constraint satisfaction problems, new impressive interactive user scenarios can be implemented inside a CAD system: like dynamic dragging of geometric elements of a solid model with preserving the initial volume of the solid body. High resolution speed required for interactive manipulation with a constrained solid model is guaranteed by LGS, which in many cases has the access to analytical representation of a system of equations and, therefore, applies advanced symbolic and numeric methods to solve it.

LGS 2D/3D Engineering Toolkit will include a specific version of LGS constraint solver extended with the possibility to solve engineering constraints (equations, inequalities and tabular relations). The interface modules for 2D profiles and 3D BRep will be extended with the possibilities to compute geometric measures curvilinear lengths, areas, volumes, centers of gravity, etc.

## Parametric Optimization

A typical scenario of parametric optimization in modern CAD systems is to choose a goal parameter (which is usually linked with a particular formula), set the direction for optimization (minimize, maximize or make it close to a goal value), select a set of free parameters (which values can be changed to reach the goal) and run a particular optimization algorithm. In this typical framework, an optimization solver treats a parametric geometric model as a black box: having no access to its analytical representation, it simply changes the values of free parameters and analyzes the change of goal parameter value. Different heuristic methods are used to optimize a block box function: hill climbing, simulated annealing, genetic algorithms, etc. This computational scheme has serious performance restriction, that's why in many CAD systems parametric optimization is considered as a hard

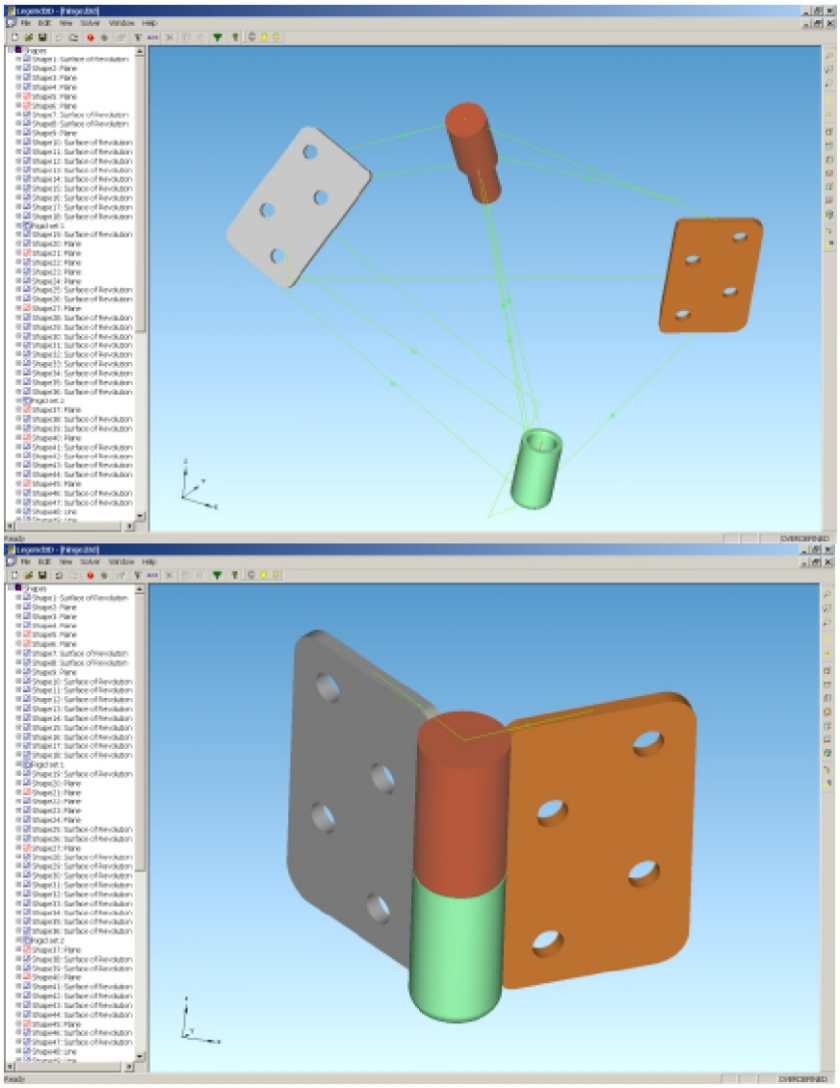

Figure 3. Botton-up assembly design based on the variational approach

task with obligatory display of a progress bar for a user, which can also stop the optimization at any time. Moreover, multi-parametric optimization of a complex 3D model shape can easily take hours or even days on modern workstations.

However, in case of implementing optimization algorithms directly inside a constraint solver, the parametric model becomes a white box, where analytical representation of most (often all) functions is known. Therefore, optimization methods of high order can be applied, where convergence speed is significantly higher.

The LGS constraint solver, dealing with mixed geometric-engineering constraint problems, is able to optimize any parameter (see Fig. 4). LGS 2D/3D Optimization Toolkit is an extension of LGS Engineering Toolkit with a set of efficient optimization algorithms.

## Kinematics Analysis

Kinematical simulation of mechanisms with movable parts is a typical application for many CAD/CAE systems. Simulated mechanism is represented as a set of rigid parts (positions of some of them are fixed in 3D space) connected with kinematical joints. A joint links a part with another part by removing some degrees of freedom from the space of their relative movements. Basic kinematical joints (revolute, prismatic, planar, spherical, cylindrical, U-joint and rigid joint) can be automatically generated from product specification taking into account the existing assembly constraints. Each joint of this kind directly corresponds to one or more geometric constraints between parts. Compound joints consist of sub-joints: gear, rack, cable, and CV-joint. Some joints (like gear, rack and screw joints) involve additional engineering specification, setting the ratio between values of translational and rotational movements of corresponding parts. Other typical joints are roll and slide joints that allow relative movement along a particular curve or surface. Many joints can be commanded it means that the remaining degrees of freedom of the corresponding parts can be used to specify their relative movement e.g., a screw joint can be commanded to simulate screw movement of linked parts.

A typical Kinematics Analysis application supports several basic user tasks: automatic assembling of a mechanism according to the given set of joints and constraints, moving its parts by a user, and automatic simulation of its kinematics according to the given user commands. Assembling and moving functionalities are similar to the ones used in Assembly Design applications, while kinematics simulation is a specific task of selecting a driving joint by the user and commanding it. An application should react to this specification by smoothly moving all parts of the mechanism according to the given set of joints, assembly constraints and user commands. Such a simulation can be performed either interactively (the user commands the driving joints with a computer mouse, while the application moves all other parts correspondingly), or in the package mode (the user first specifies a law of motion for the driving joint of the mechanism by writing down a formula for changing its parameter in time, while the application computes the corresponding movement of all other parts and then visualizes the movement as a movie).
More complex user scenarios include kinematical analysis of a mechanism (detecting its internal degrees of freedom, measuring movement speed and acceleration of a particular part, displaying the trajectories of the movement, computing minimal distance between parts during the movement, detecting collisions between parts, computing swept volume for a movement of a given part, etc.)

Despite the similarity of kinematics analysis to assembly design, the variational functionality required in these applications is different. First of all, a constraint solver should support not only geometric and engineering constraints, but also the joints themselves. Also there should be a possibility to specify commands for driving joints (including specifications of a law of motion). LGS 3D Kinematics Toolkit will include the extended version of the LGS geometric constraint solver supporting the above-described functionality.

## Rigid Bodies Dynamic Simulation

Dynamic simulation of rigid bodies' motion available in many CAE systems is based on the analysis of physical forces affecting them (gravity, supporting force, friction, etc.), taking into account collisions between their parts and assembly constraints (joints) imposed on them. As a rule, a user of a typical dynamic simulation application has the ability to import a geometric model of an assembled mechanism (or its parts) from a CAD system (often with BOM specification, since physical properties of a part are detected from both its geometric shape and its material), to set constraints and joints between its parts, and to set physical forces, which can affect either the whole body, or on some parts of its surface, or at some specific points. After that the application automatically computes mass and inertia tensor of each part, simulates the motion of the mechanism in the given time interval, and visualizes it as a movie.

A well-known approach to simulation of rigid bodies' dynamics consists of modeling their motion locally with Newton-Euler ordinary differential equations. (Global model of motion of a mechanism should take into account each constraint/joint and each collision between parts; they are resolved into additional physical forces acting on them.) However, such a modeling is not well suited for large industrial mechanisms since the dimension of the corresponding system of ordinary differential equations can easily counteract the ability of modern workstations to solve it with iterative methods of numerical analysis. Fortunately, Lagrangian mechanics allows a more compact representation of motion with Euler-Lagrange equation, where generalized coordinates can be used to represent the motion of each part. Using generalized coordinates only for remaining degrees of freedom of each part (since joints and constraints remove many of them) one can significantly reduce the dimension of a system of differential equations of motion. This principle will be used to build LGS 3D Dynamics Toolkit a set of libraries for compact modeling of motion, computing mass and inertia tensor, and detecting and resolving collisions between rigid bodies in boundary representation.

## Direct Geometry Editing

Translating a geometric model from one CAD system to another with standard CAD data translation tools usually means losing any parametric information related to the model. Advanced feature-based translation and feature recognition tools are able to translate some design features, but not all of them, since the sets of available features and sets of their parameters usually differ significantly in two particular CAD systems. It means that an imported geometry usually is fully or partially "dumb", i.e. is not parametrically editable (as it was in the original CAD system). It constitutes serious challenge for CAD developers, since global cooperation between engineering and manufacturing companies worldwide involves a lot of different CAD systems and CAD data formats. The real break-through in this field has been achieved only recently with the introduction of a new conception  direct geometry editing. With this approach, a "dumb" geometric model (without features) available in its boundary representation (BRep) can be user-edited by dragging its topological entities and setting geometric constraint between them. Topological elements of BRep are its vertices, edges and faces. Each topological element is based on a geometric one (vertices are points, edges are fragments of lines or curves, while faces are fragments of planar and curvilinear surfaces). Thus, one can easily represent a particular BRep model as a set of geometric elements connected with topological constraints (each edge is incident to two vertices, each face is incident to several edges and vertices). Additional logical and dimensional constraints can be added to this model, specifying, for example, the tangency between two particular faces, angle between two edges, distance between two vertices, radius of a cylindrical or spherical surfaces, etc.
Therefore, it is easy to build an application based on variational constraint solvers like LGS, which allows 3D editing operations similar to the ones used in 2D sketcher: setting new constraints, removing an old one, changing the values of parametric constraints, and dragging geometric elements while keeping all imposed constraints satisfied. Moreover, this powerful new approach to design allows its applications outside CAD data translation domain. Indeed, direct geometry editing capabilities could supplement traditional 3D modeling tools. All traditional advantages of variational approach  setting as many constraints as needed, order of constraints has no importance, dynamic dragging under constraints  minimize design time and maximize creativity.

LGS 3D Interoperability Toolkit will include a subset of basic 3D functionality of the LGS geometric constraint solver as well as additional modules: BRep API module allows setting constraints directly on topological elements and automatic creation of coincidence constraints from topologically adjacent elements; constraint recognition module detects obvious logical constraints between edges and faces, while integration module simplifies integration with popular geometric modeling kernels.

## Configuration Design

Designing parts and mechanisms from scratch is a very creative and time-intensive process, requiring advanced CAD tools. However, often it is needed to solve very simple design problems, like assembling a product from a set of standard parametric parts. Typical individual products

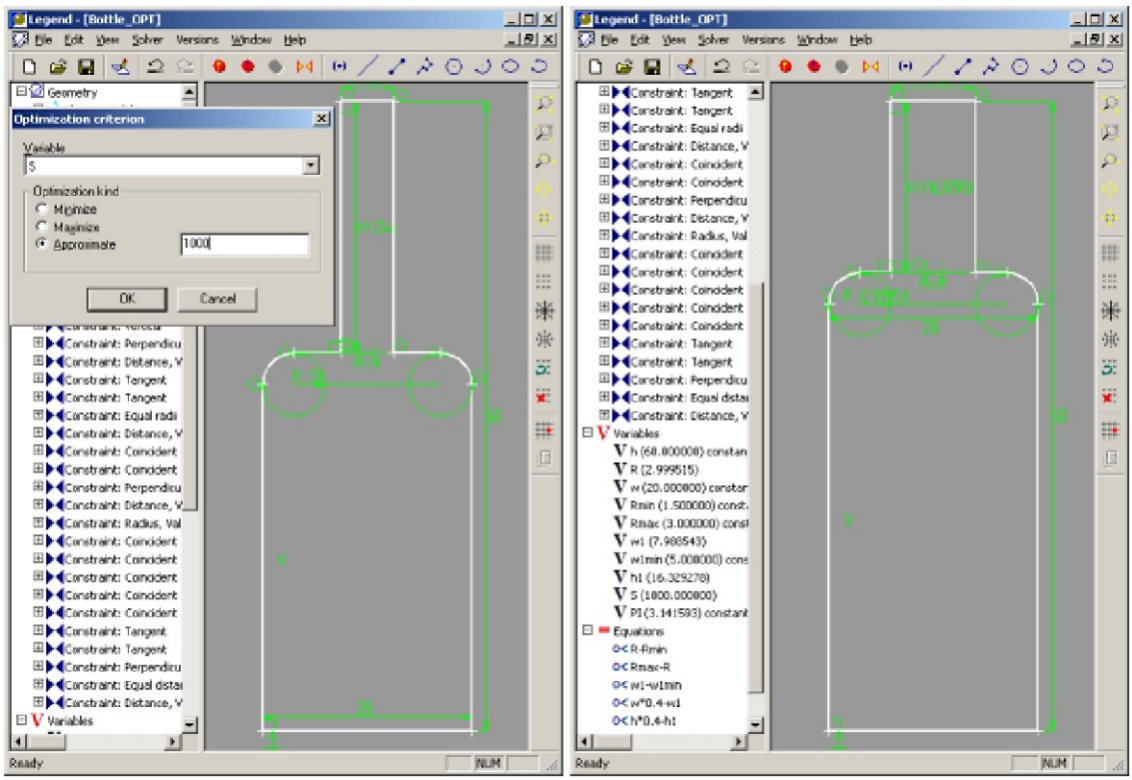


Figure 4. Closed contour inner area optimization with LGS

composed of standard parts are windows, doors, stairs, kitchen furniture, swimming pool, etc. Designing such a product, one has to instantiate some 3D parts from a given catalog by setting required dimensions to them and to assemble the parts together. Usually it is not economically feasible to use a standard 3D CAD system for such a design, since no 3D modeling or mechanical assembly tools are needed here. That's why many furniture vendors develop their own simplified pseudo-3D applications to create a standard product for a particular (individual) customer, to compute its price and to form an order for its manufacturing.

LGS 3D Configuration Toolkit will include all functionalities needed to develop simple 3D CAD applications for creating products from a set of standard parametric parts. It will be based on a very limited version of LGS 3D, accessible at affordable price.

### 3D Viewer

Design and manufacturing data exchange in extended enterprise (across a supply chain) is not possible without simple and handy tools for visualization of 3D models. Usually these tools can be easily integrated into Internet browsers and office applications. They are based on compact (rough) representation of a geometric model and have only basic features of 3D navigation (magnification/demagnification and rotation/translation) as well as the ability to play 3D movies (usually needed for visual explanation of maintenance and servicing operations related to a particular product). Next logical step in extension of functionality of such simple 3D viewers is to allow their users to see and play with internal degrees of freedom of mechanisms. For example, a user can not only see a window from different angles, but also virtually open and close it.

A dedicated "light" version of geometric constraint solver LGS will be included into LGS 3D Viewing Toolkit, a set of software modules for rapid development of 3D visualization applications with the possibility to play with internal degrees of freedom.

### Beyond Mechanical CAD Applications

Applications of variational solver LGS with its advanced features for geometric and engineering constraint satisfaction and optimization are not limited to mechanical CAD systems, discussed above. Below we outline other directions, where variational design approach adds value to the existing applications.

Education is one of such prospective field. FlashLGS is an example of educational application, which is accessible directly from any Internet browser. It combines Adobe Flash® technology with variational constraint solver LGS 2D (see Fig. 5). The application allows creating basic geometric elements (points, linear segments and circles) and linking them in an arbitrary order with logical and dimensional constraints. Its intuitive measuring GUI allows to study the variational design concepts in just a few minutes without any preliminary training. Moreover, this or similar application can be used at schools to create simple illustrations for planimetric theorems and problems.

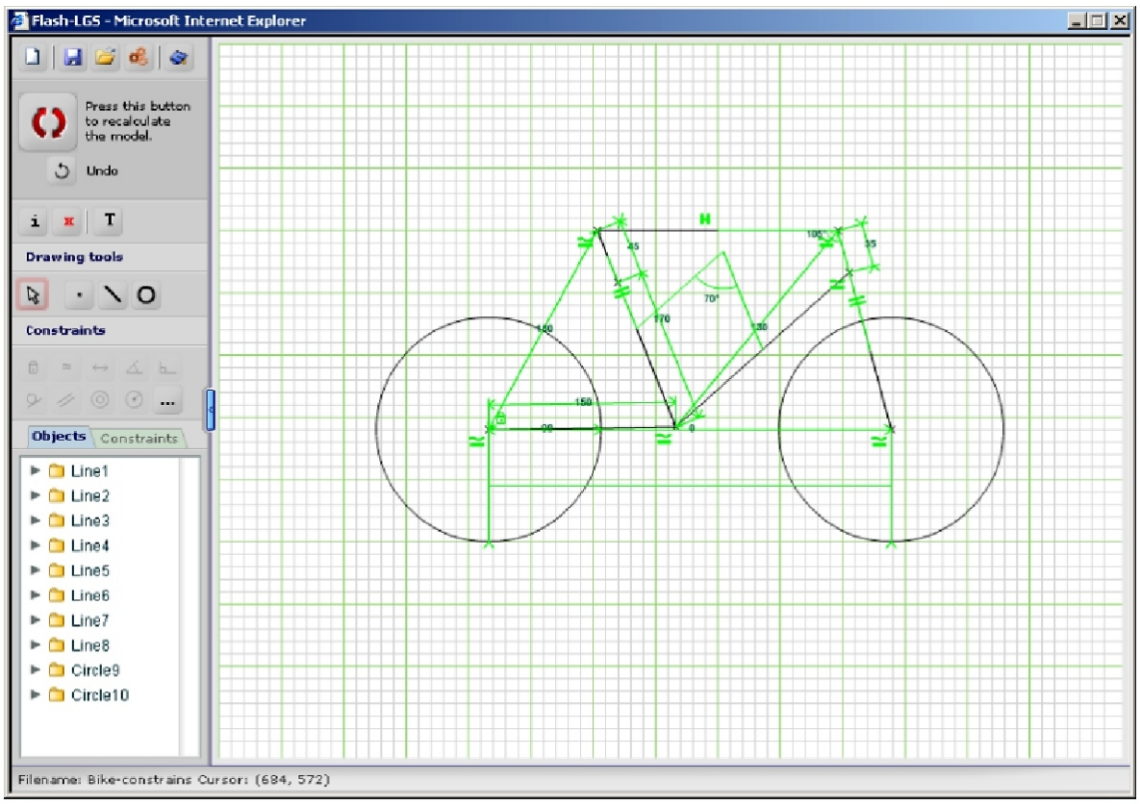


Figure 5. Variational sketching inside Internet browser (FlashLGS)

CAD 4D is a popular concept in AEC (Architecture Engineering and Construction) field allowing visual planning of construction process. Here a designed 3D model is changed in time in accordance with a given construction schedule (3D + Time = 4D). Assembly constraints connecting parts can be directly associated with resource constraints, while process of 3D model modification is associated with a construction tasks schedule. It is important to note, that resource scheduling is another fruitful application area of variational constraint approach. Moreover, LEDAS provides a dedicated constraint solver called Scheduler, which is able to satisfy typical scheduling constraints (precedence of tasks, usage of same resources, etc.) and to compute an optimal (w.r.t. given criteria) schedule. This technology can be integrated with LGS to form a powerful computational platform for developers of CAD 4D applications.

Computer vision is an important application field for geometric constraint solving. The main problem here concerns reconstructing a 3D model from a set of its 2D images. Considering these images as projections of a 3D body, one can form a variational model where each 3D element has its 2D projection (it can be easily expressed in geometric terms), and constraint satisfaction here consists of putting 3D elements in such a position, where all projection constraints are satisfied.

Computational biology (bioinformatics) models different interactions between macromolecules with important applications in such fields as medicine and pharmacology (drug design), agricultural industry (fertilizer design), or nanotechlogy. One of the most important problems here is protein structure prediction aimed at computing the 3D structure of proteins from their amino acid sequences. A popular method to solve this problem (known as comparative modeling) uses Protein Data Bank (PDB), which stores already known 3D structures of simple proteins. After matching similar subsequences of amino acids, one then has to compute an initial 3D model, where Cartesian coordinates for atoms in the protein are computed from satisfaction of spatial restraints (distances and angles with probability density functions). This model serves as the basis for a global optimization procedure, which iteratively refines the 3D positions. Geometric constraint solver LGS 3D can be used here to satisfy (most of) distance and angle constraints and to compute the 3D coordinates. Practical experiments showed the ability of LGS to solve problems with dozens thousand atoms very efficiently.

**About the Author**

Dr. Dmitry Ushakov, Chief Technology Officer at LEDAS Ltd., graduated from Novosibirsk State University in 1995 with a Master's degree in Mathematics. In 1998 he was awarded a PhD in Computer Science at the same university. In 1993-1999 Dmitry worked for the Russian Research Institute for Artificial Intelligence. His interests were in the area of theoretical justification of the apparatus of subdefinite models (a kind of constraint programming) and development of the NeMo+ object-oriented constraint programming environment for solving a broad range of constraint satisfaction problems in various application fields. Since 1999, Dmitry has been on the staff of LEDAS. Until 2001, he was responsible for the cooperation with Dassault Systemes, a world leader in 3D and Product Lifecycle Management solutions. During his two-year stay in France, he developed several knowledgeware components based on the variational constraint solver for CATIA V5. Dmitry Ushakov is the author of 50 scientific papers.

**About LEDAS**

LEDAS Ltd. is an independent software development company founded in 1999; it is based in Novosibirsk Scientific Centre (Akademgorodok), Siberian Branch of the Russian Academy of Science. A leader in constraint-based technologies, LEDAS is a well-known provider of computational software components for PLM (Product Lifecycle Management) and ERP (Enterprise Resource Planning) systems: geometric constraint solvers for CAD/CAM/CAE, optimization engines for Project Management, Work Scheduling and Meeting Planning as well as interval technologies for Knowledge-Based Engineering and Collaborative Design.
The company also provides services for PLM and ERP markets: software development, consulting, reselling as well as education and training. More information about LEDAS is available on the Internet at:http://ledas.com