# Development of DeltaFix Tool for Fixture Synthesis in NX 10.0 Platform, Based on C++ and NXOpen API Library

## (Instruction manual)

使

用

说

明

书

Created by: Taqiaden Alshameri

## *Table of Contents*

# 1. Introduction

Computer technologies have revolutionized the way products are manufactured today. Routine manufacturing tasks are being distinguished analyzed and computerized. This leads to great reduction in the manufacturing cost, time and the demand for high expertise to perform engineering tasks. Non-experts are now enabled to perform complex operations with minimum interaction with the control parameters. As part of this revolution, a new category of tools called computer-Aided Fixture Design (CAFD) has emerged to incorporate fixture design functions into CAD environments. A conventional CAFD includes packages for fixture synthesis, fixture analysis and fixture verification.

Fixture plays essential role during any machining process, in fact, the precision capability and the accuracy of the produced parts are highly affected by the robustness of the workpiece-fixture system. The main function of fixtures is to hold and secure machine part in place during machining. Fixture design and fabrication can utilize a considerable amount of the total cost of a manufacturing system. Automating and accelerating the process of selecting fixture layout can mitigate much of the complexity associated with fixturing problem.

In this context, DeltaFix tool was developed to handle the task of robust fixels arrangement on the workpiece. And by fixels, we mean either a locator or a clamp. The main aim of the tool is to determine the best allocation of fixels on the workpiece surfaces to meet the robustness criteria. The scope of the tool is to constrain a 2-D workpiece by a number of three locators and two clamps (Figure 1). Three locators are sufficient for deterministic locating of the workpiece and with two clamps we can hold and secure the workpiece in place during the machining process.

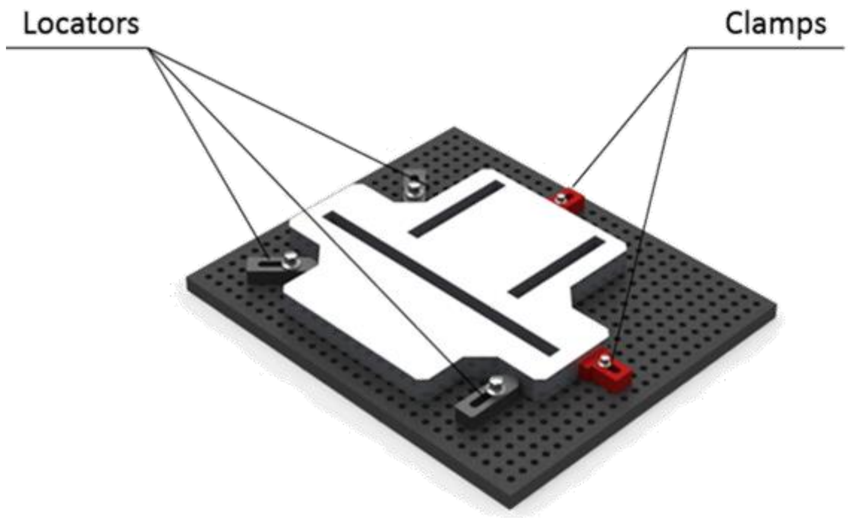


**Figure 1 Fixture layout for 2-D workpiece**

DeltaFix tool is targeting the NX 10.0 CAD platform as an operational dependency. NX 10.0 platform has empowered third party developers to build customize tools by calling generic inner function of NX. NX Block UI styler application is used to build a dialog box to manage the user interaction with the CAD environement. NXOpen libraries are the backstage players that we rely on to communicate with the Cad model in NX.

In brief, DeltaFix is an external tool working on NX which is developed using C++ and NXOpen libraries. The aim of the tool is to automate the selection of the optimum fixture layout in a CAD environment for arbitrary 2-D workpiece. The subsequent chapters of this manual provide guidelines and instructions for using DeltaFix tool and a bit overview of the working principles.

## 2. Building and operation environment

The DeltaFix tool can be invoked by calling the tool DLL file from the NX 10.0 environment by selecting (File->Execute->NX Open…), then, a dialog box will appear to choose the “DeltaFix.dll” file.

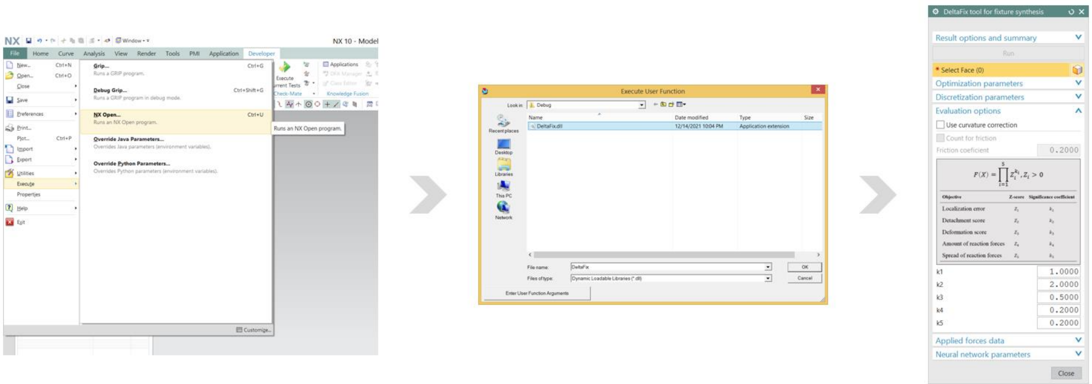

**Figure 2 Launch DeltaFix tool**

For user convenience, all Input and output fields are organized in one dialog box and organized in six group layouts. The tool interface follows the same standard as other NX classic interface so the user can rabidly get familiar with it.

One important point to mention is that the DeltaFix tool is compatible with version 10.0 of NX, hence using different versions of NX to run the tool may confront problems.

The default tool directory for retrieving DLX and data files is at “…\DeltaFix tool files \” with the same driver prefix of dll file directory of the tool. DLX is a requested file which used to store the UI layout and is an essential part to initiate the tool. For any exported outputs, the default directory is at “…\DeltaFix tool files\output\”.

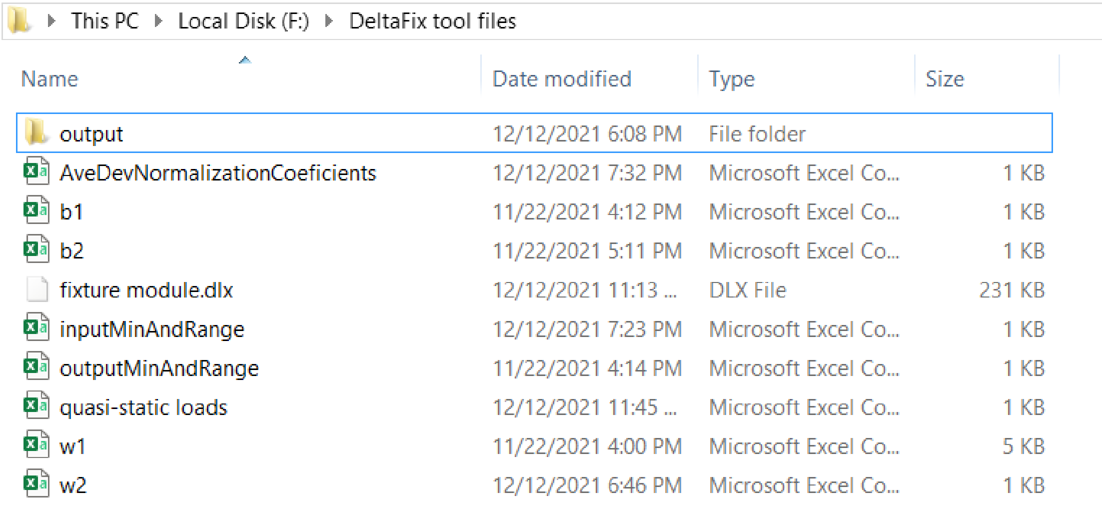


**Figure 3 Attached files directory**

Despite previous versions of the tool, the current state of the tool is made independent of the CAD model and no NX part templates are required. A user can apply the tool function directly on any model part. This feature has made possible by removing the stickiness of the tool to a specific CAD model; DeltaFix tool has an inner transformation mechanism to deal with different orientation of the model. Moreover, the selected entity is recognized and transformed into a generic form of the problem.

# 3. User Interface

The main window (Figure 4) of the DeltaFix tool composes of six groups of input/output fields (Figure 5), a Progress bar, a trigger button and a face selection button. The progress bar shows the progress state during the execution of the optimization process. The trigger button is enabled once a face entity has been selected. The six groups are detailed below.

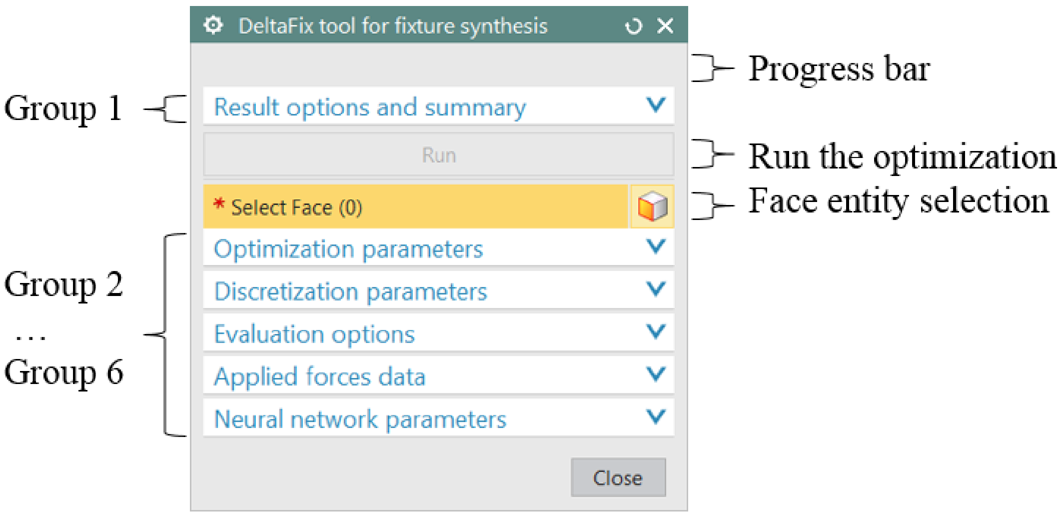


**Figure 4 DeltaFix main window**

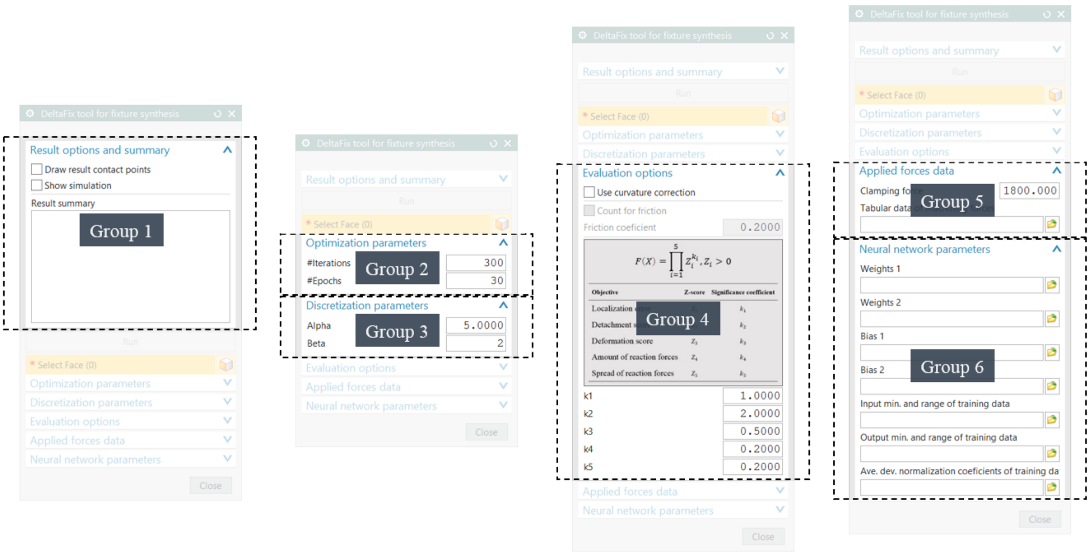


**Figure 5 DeltaFix group layouts**

## 3.1 Result options and summary

This group has two toggle buttons to specify the user preferences to view the result. The first toggle (Draw result contact points) will draw points entities at the end of optimization task to indicate each fixel contact point of the optimum fixture layout. The second toggle (show simulation) will demonstrate to the user the simulation progress at some intervals during the optimization process. At the end the, a report of the final result will be shown in the result summary multi-line box.

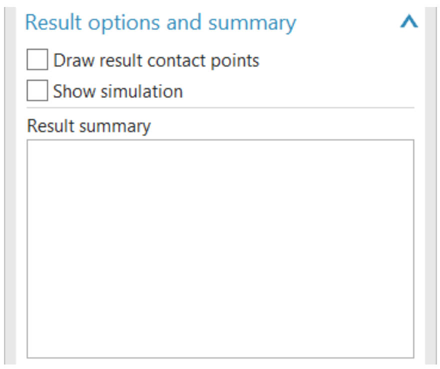


**Figure 6 Group 1: Result options and summary**

## 3.2 Optimization parameters

The number of iterations and epochs can be altered in this group. The recommended values are as shown in Figure 7. According to the neighborhood simulated annealing algorithm the optimization temperature and the size of the selection pool is altered at an outer loop. The inner loop in the other hand, evaluate a number of candidates generated at constant temperature and chosen from the same selection pool. The number of iterations determines the repetition of the outer loop where the number of epochs determine the repetition of the inner loop.


**Figure 7 Group 2: Optimization parameters**

## 3.3 Discretization parameters

Within the process of fixture layout optimization, the face entity is discretized with a step size alpha. A small alpha value holds high accuracy in computing the mathematical attributes, however, a smaller value of alpha is better in view of computational burden. A role of the thump is to specify a value for the alpha equal to 1/100 the maximum circle diameter that can be drawn within the workpiece boundaries.

Beta is the pattern number for excluding points result from discretizing the face entity from the selection pool of candidates. Similar to the case of alpha value, high beta values can accelerate the optimization process but this put the risk of omitting valuable candidates from the selection pool.

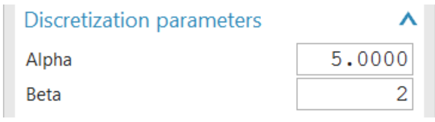


**Figure 8 Group 3: Discretization parameters**

## 3.4 Evaluation options

Placing a fixel at high curved region on the workpiece surface can introduce more localization error. Therefore, a curvature correction option is included to quantify this error when evaluating candidates. Only workpieces of curved surfaces can benefit from this option.

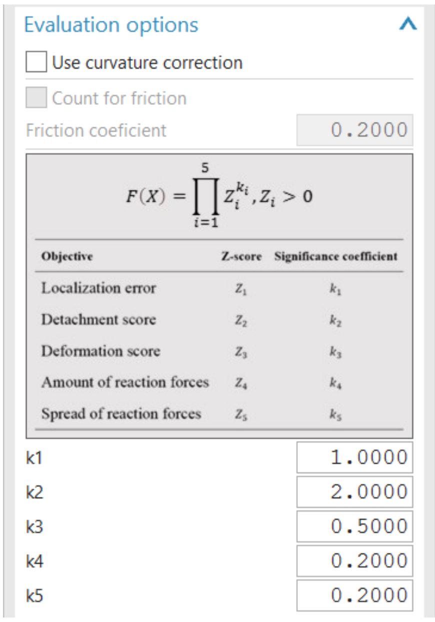


**Figure 9 Group 4: Evaluation options**

The objective function is a product combination of five sub objectives given by the following equation:

$$F(X) = \prod_{1=1}^{5} Z_i^{k_i}, \forall\, Z_i > 0$$

$Z_i$ is the objective z-score shifted to the right i.e., the average is not zero (as in normal z-score calculation) but shifted to the positive coordinate so all z-scores values are maintained positive. The significance of each objective can be determined by the significance coefficient $k_i$ which is determined based on the engineering purpose. A default value is set to $(k_1, k_2, k_3, k_4, k_5) = (1.0\,, 2.0\,, 0.5\,, 0.2\,, 0.2)$. More Detailed explanation of the significance coefficient and the associated sub objectives are shown in Table 1.

**Table 1 Significance coefficients and the associated sub objectives**

| Significance coefficient | Objective | Description |
|---|---|---|
| $k_1$ | Localization error | Errors in Locators-workpiece contact points (known as localization source) contribute to the formation of workpiece positional errors. Positional errors are measurement errors in the machined features of strict tolerances requirements. This error is mainly induced by the misalignment at locators-workpiece contact points and the relative micro-scale movement between workpiece and locators during the machining process. |
| $k_2$ | Detachment score | The workpiece-fixtures system is treated as quasi-static and machining forces are presented at discrete intervals during the machining process. Inappropriate allocation of clamping points can cause undesirable detachment of the workpiece-locator contact during machining. |
| $k_3$ | Deformation score | Excessive clamping and cutting forces can affect the accuracy of the machining process. Moreover, it can add more risk to workpiece breaking or yielding. |
| $k_4$ | Amount of reaction forces | In some occasions, Higher reaction forces are desirable to increase the stability and dynamic resistance of the workpiece during machining. |
| $k_5$ | Spread of reaction forces | Maintain equal distribution of reaction forces at each locating point is preferable, so that all fixel elements produce the same amount of reaction force. |

## 3.5 Applied forces data

The clamping and machining forces are specified at this group relative to the absolute coordinate system of the NX model. The tabular data of machining forces are in the same form as shown in Table 2 but without the heading names. The table of pure data entries must be in CSV format. The $F_{x,i}$ , $F_{y,i}$ , $F_{z,i}$ are the force components, where, $x_i$ , $y_i$ , $z_i$ are the coordinate of the force application point. As stated previously, the DeltaFix tool applies the roles of quasi-static process and hence the loads in the table are estimating at discrete intervals, and each interval is assigned an exposure coefficient ($EC_i$) to indicate the amount of time a load has been applied on the workpiece. For equal intervals, all $EC_i$ are the same.

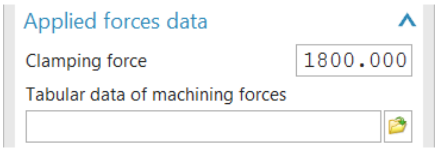


**Figure 10 Group 5: Applied forces data**

**Table 2 Machining forces tabular form**

| $F_{x,i}$ | $F_{y,i}$ | $F_{z,i}$ | $x_i$ | $y_i$ | $z_i$ | $EC_i$ |
|---|---|---|---|---|---|---|
| | | | $i = 1$ | | | |
| | | | … | | | |
| | | | $i = n$ | | | |

## 3.6 Neural network parameters

Deltafix tool incorporate the effect of the workpiece deformation during the evaluation of candidates (sub objective 3). A common technique to estimate the deformation level of a workpiece under a given load is by using a Finite Element Analysis (FEA). Though, relying on this method is computationally expensive. To overcome this drawback, the DeltaFix tool employ a Neural Network (NN) model to substitute the FEA method. The NN model is trained using data acquired from the FEA analysis of different workpieces at different clamping and machining force conditions. The Gained computational efficiency after adopting the NN evaluation method is tremendous compared to FEA.

This group layout asks the files for weights, biases, normalizations and standardization coefficient. The files are in CSV format and are already equipped with the tool. By default the tool looks for these files at "…\DeltaFix tool files \" directory. If not exist, the user must specify the exact location of these data files.

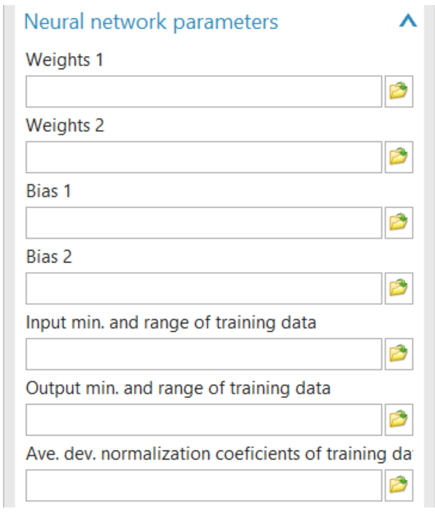


Figure 11 Group 6: Neural network parameters

# 4. Theoretical basis of DeltaFix tool

DeltaFix tool has been developed base on original work that contains two main aspects of novelty; For the first aspect, A new optimization algorithm has been used to carry out the optimization task called Declining Neighborhood Simulated Annealing Algorithm (DNSA) which was developed specifically for the fixture synthesis problem. For the second aspect, A new multi objective approach is adopted to carry out the evaluation process of potential candidates.

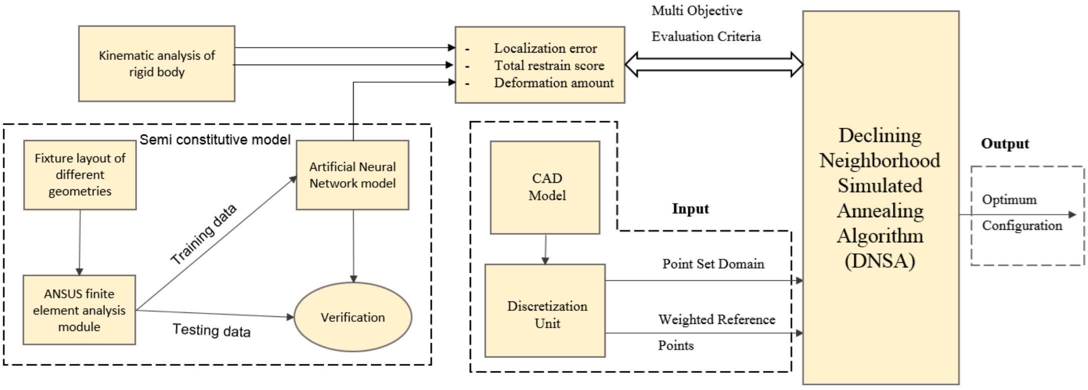


**Figure 12 DeltaFix tool operation framework**

The framework of DeltaFix tool is illustrated in Figure 12. Applying the DNSA algorithm with multi objective for fixture synthesis in a Computer-Aided Design (CAD) environment is the main purpose of the tool. The first two objectives of the sub objectives are extracted from the rigid body fixture model results from kinematic analysis of workpiece-fixels system. The third objective is dedicated to study the deformation behavior of the workpiece. This have been achieved by integrating FEA and ANN. The last two objectives are set to satisfy locator reaction forces requirements. The inputs to the DNSA are point-set domain and weighted reference points. Point-set domain is simply discrete points (extracted in the CAD software NX) of all possible coordinates on workpiece surfaces that are feasible for locator placement. Weighted Reference points are essential for the evaluation process. The algorithm output is expected to be equal or sufficiently close to the global optimum.

## 4.1 Optimization algorithm (DNSA)

A heuristic approach is capable of solving the complex multi convex problem of fixture synthesis. DNSA algorithm is a novel heuristic which is a variant of the original algorithm, referred as Standard Simulated Annealing (SSA). The new algorithm is developed specifically for the current problem of fixture synthesis. The proposed algorithm differs from the SSA in two aspects: First, DNSA exploits a set of previously accepted candidates to predict the next move. Second, the generation mechanism of new candidates is governed by a declining probability density function (PDF). To put it differently, a group that keeps records of the last accepted candidates is employed to direct the search area

## 4.2 Novel multi objective evaluation schema

To fulfill the criteria of robust selection of fixture layout, multi sub objectives are involved. These objectives are summarized in Table 1. All of these objectives except for the third one are derived from kinematic analysis and rigid body workpiece model. The third objective is derived from elastic workpiece model which is basically a neural network model (see section 3.6). The NN model substitutes the common practice of using FEA models, the later hold high computational cost and is unpractically to be integrated in our tool, In the other hand the former model is computationally efficiency and cause no significance delay when evaluating the deformation during the run of the DNSA optimization.

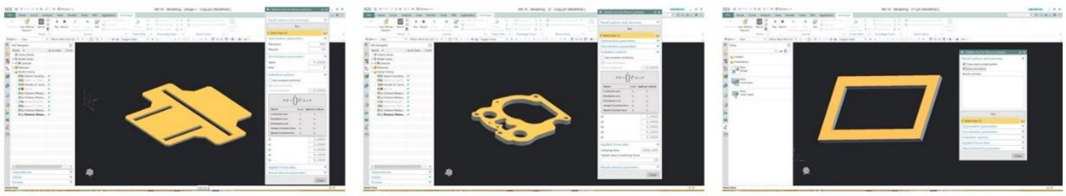

**Figure 13 Flexibility of DeltaFix tool**

# 5. DeltaFix process flow

The main working flow after a user have pressed the “run” button is as follow:

## 5.1 Initialization

- Retrieving all data set by the user.
- Clear variables values from previous run.
- Load all tabular data of the neural network and quasi-static loads.
- Initialize inner variables.

## 5.2 Transform axes

- Transform the axis of the face entity selected by the user to a 2d $x, y$ axis.
- This step is inverted at the end of the optimization run to get the original coordinate of the result.

## 5.3 preform discretization

- Convert the selected face entity to a set of uniformly spaced points, with $\alpha$ being the step-size.
- Isolate points in the boundary to a separate container.
- Determine the reference points.

## 5.4 Compute mathematical attributes

- This includes estimating the first derivative, second derivative, curvature, and norms for each point in the outer boundary.

- Part of the neural network input variables are computed in this step such as points intensity in small and big region, points in forward beam and shear line.

## 5.5 Extract selection domain

- Feasible candidate points at the boundary are spotted and kept in a separate container.
- These points will be the selection pool when running the optimization task.
- Points at high singularity region are spotted in this step and excluded from the selection domain.

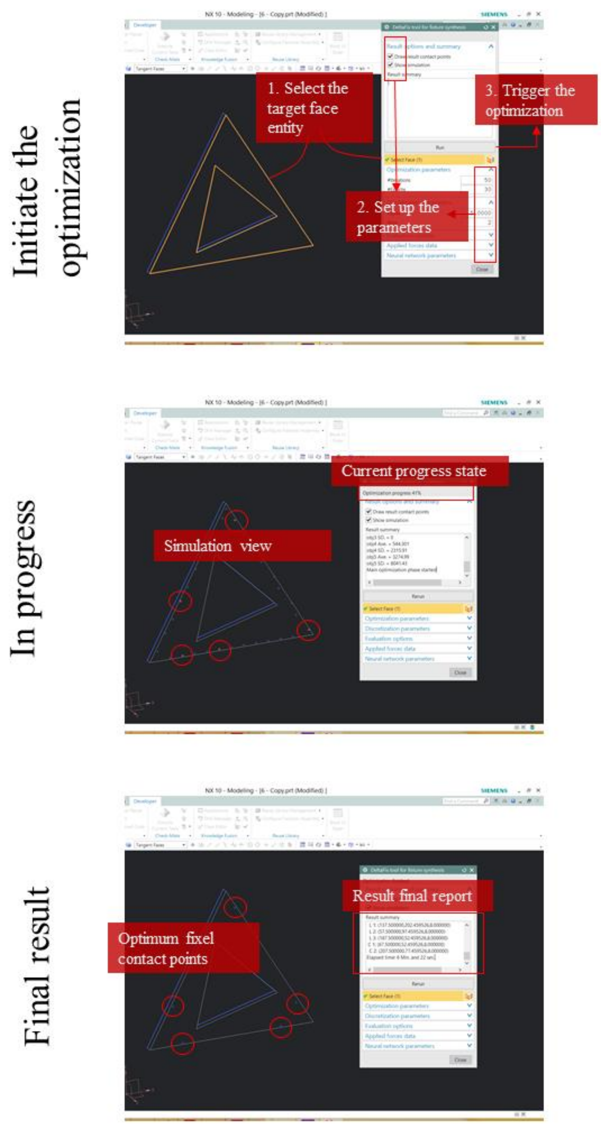


**Figure 14 DeltaFix tool in action**

## 5.6 Start DNSA optimization

- This step includes running the Declining neighborhood simulation algorithm (DNSA) two times.
- First run of the DNSA is fast and is aimed to calibrate the multi objective equation.
- Second run is the real optimization task which takes a bit longer.

## 5.7 Report the result summary

- A summary of the optimization process is concluded in this step.
- The optimum positions of the locator and clamps are specified.
- A txt file is exported showing the progress of the optimization task. Errors will be reported if any.

# 6. Software project engineering framework

The main structure of DeltaFix tool is shown in Figure 15. The internal dependency of the tool is categorized into four layers, namely; “fixturemodule”, “optimizationengine”, “nxbasicfunctions”, and “calculations”. The external dependency of the tool is either a generic C++ library or an NXOpen interface library. Figure 17 illustrates the main call of the external classes and methods.

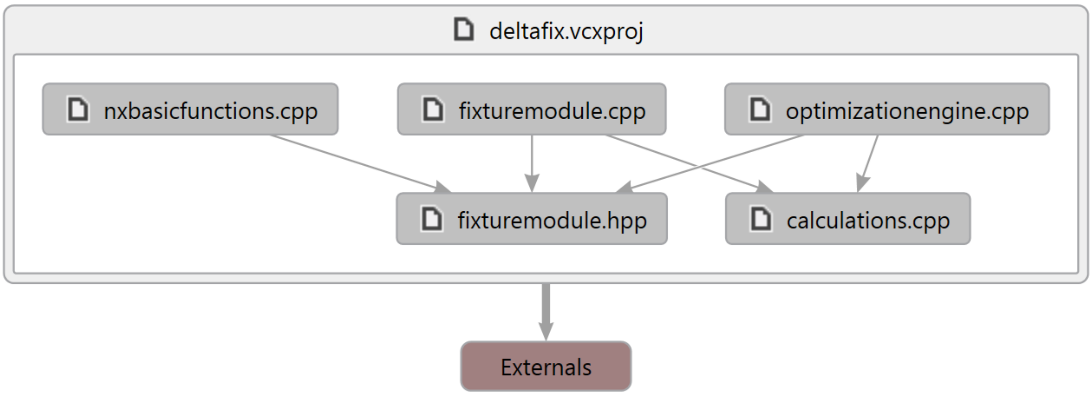


**Figure 15 Dependency diagram**

## 6.1 First layer (fixturemodule.cpp and fixturemodule.hpp)

This is the main layer which manage the interaction between all layers. For convenience and readability of the code, the headers in this layer are separated to one file “fixturemodule.hpp”.

The main method of DeltaFix tool exist in this layer which called “runDNSA”. This methods is handles all the execution flow of fixture layout optimization. The main dependencies of “runDNSA” method is shown in Figure 16.

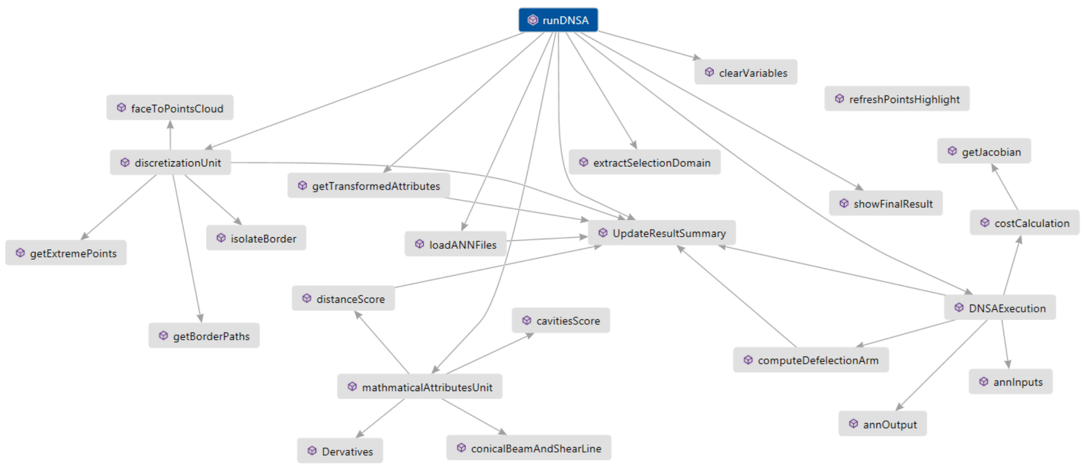


**Figure 16 The main method of DeltaFix tool**

## 6.2 Second layer (optimizationengine.cpp)

The development of DeltaFix tool includes original work of optimization mechanism and multi objective evaluation process. All operation related either to any of these two aspects are organized in one class layer called “optimizationengine.cpp”.

## 6.3 Third layer (“nxbasicfunctions.cpp”)

The interaction with the NX Cad platform is coped with in this layer. Basically, This layer composes of NXOpen based methods which are invoked to serve different communication purposed with the CAD model. For instance, to check if a point lay on a face entity, then, the following function should be called “UF_MODL_ask_point_containment()” and encbsulated within “UF_initialize()” and “UF_terminate()” methods to allocate and deallocate a UG/Open API license respectively.

## 6.4 Fourth layer (“calculations.cpp)

The basic mathematical operations are performed in this layer. For instance, the computation of standard deviation, derivatives, curvature, probability density function generation mechanism, etc.

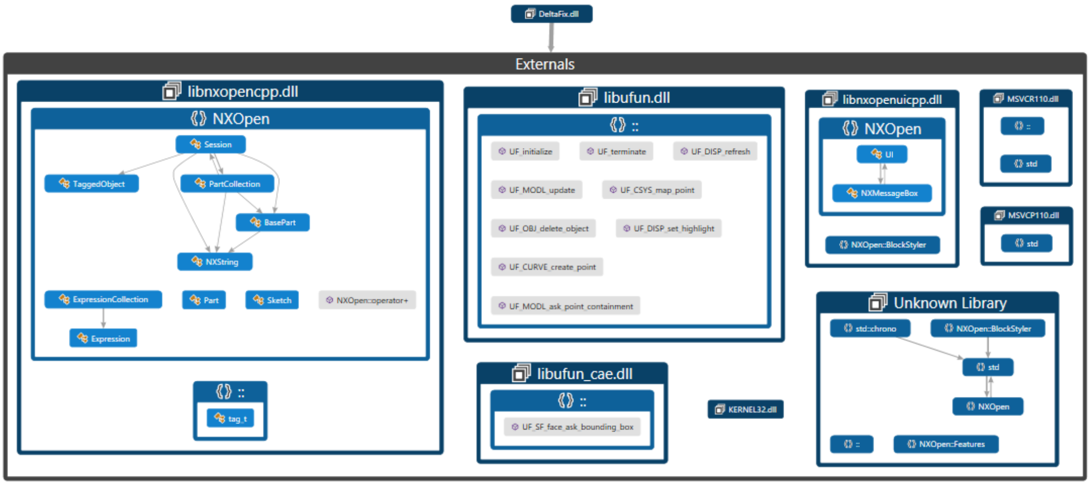


**Figure 17 Solution external dependencies**