Министерство образования Республики Беларусь

Белорусский национальный технический университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчёт**

по лабораторной работе № 7

по дисциплине ***«Методы и алгоритмы компьютерной графики»***

Вариант 4

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | ст.гр. Кишкурно М.В. |
| Преподаватель: |  | ст.пр. Тетерюкова И.О. |

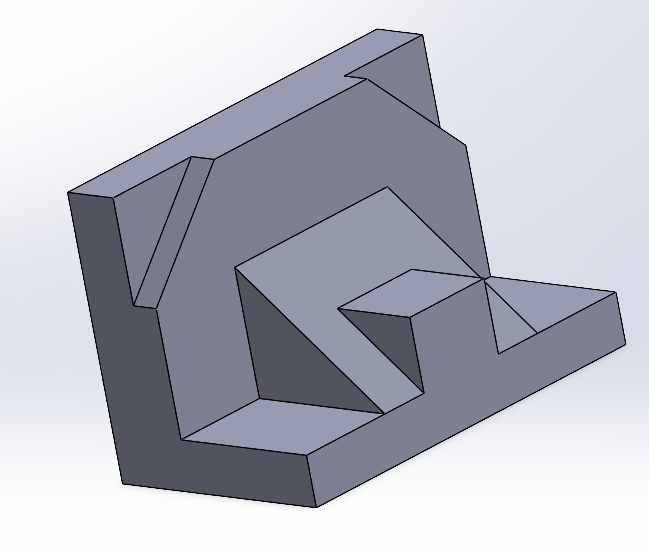
2023

**Цель работы:** Получить навыки в процессе проведения инженерного анализа деталей с помощью SW API.

**Задание:**

Для 3D модели детали, провести инженерный анализ в интерактивном режиме.

Вариант задания 3D модели соответствует варианту, выполнения лабораторной работы по 3D конструированию детали.

****

**Результаты выполнения работы**

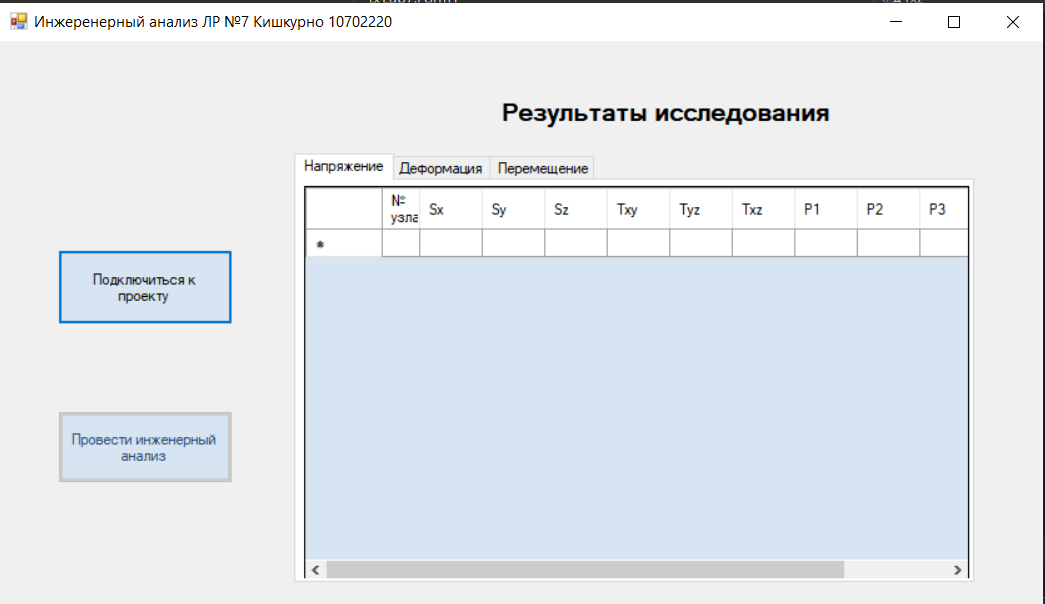
****

Рис.1 – Окно запуска программы

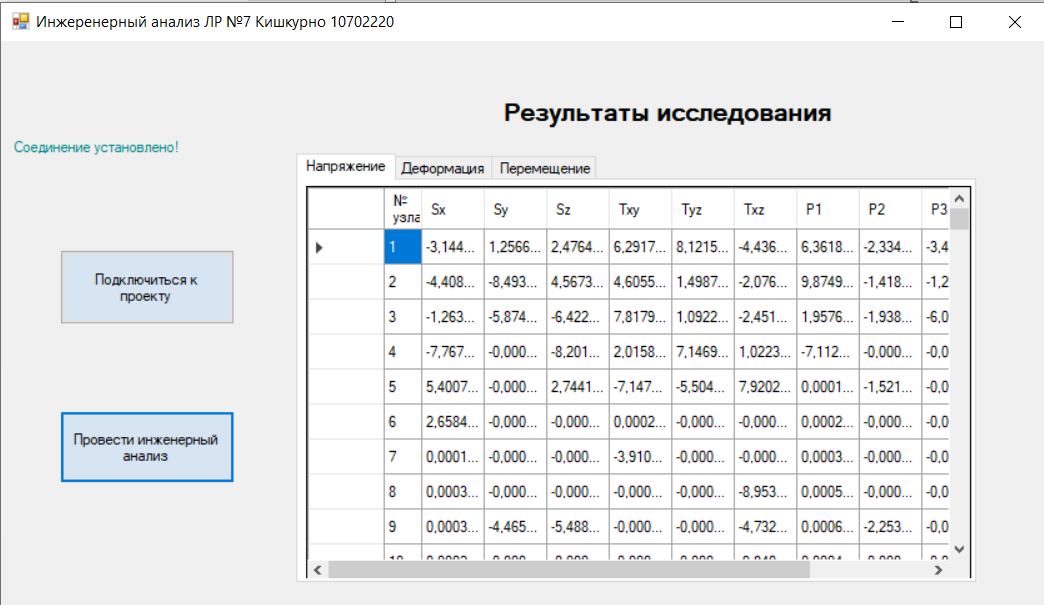
****

Рис.2 – Результаты исследования на напряжение

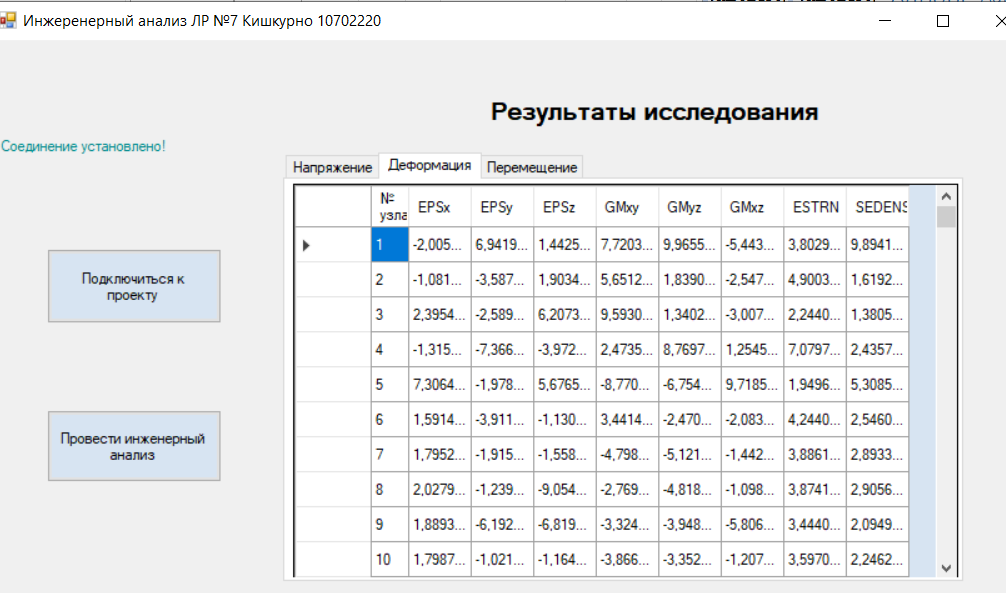
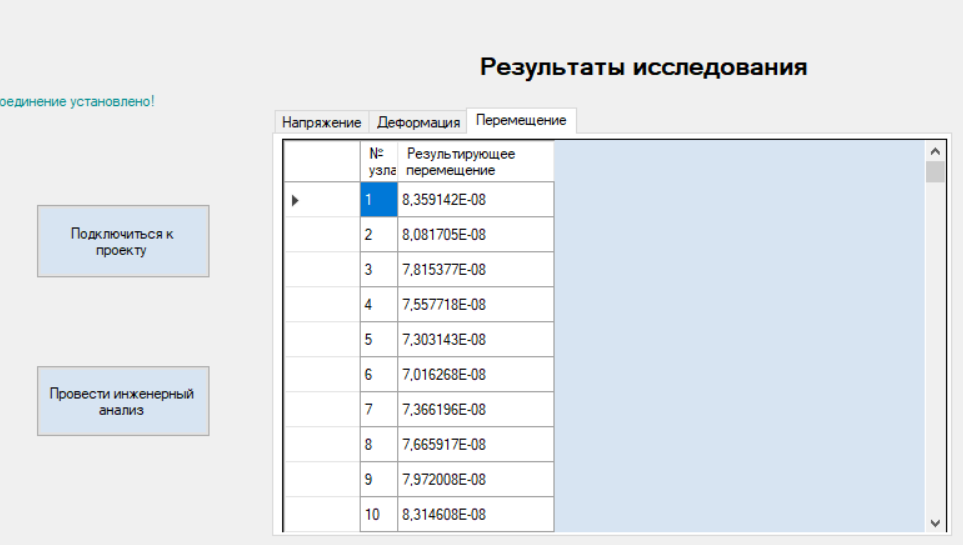
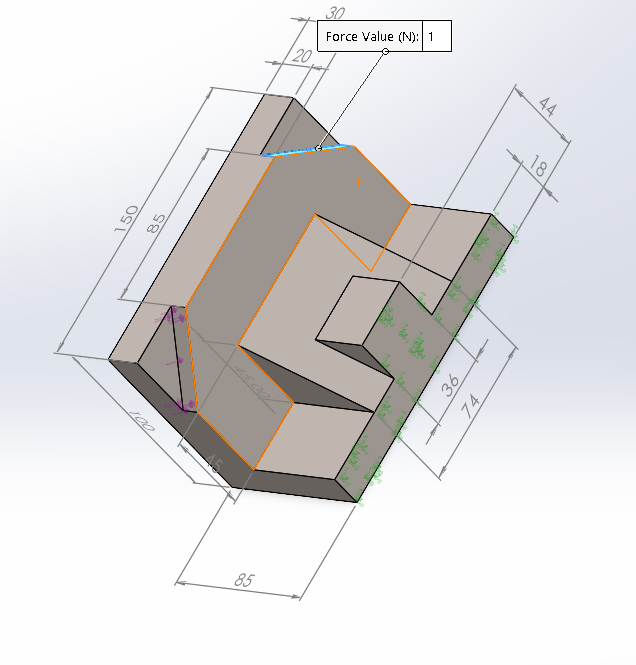
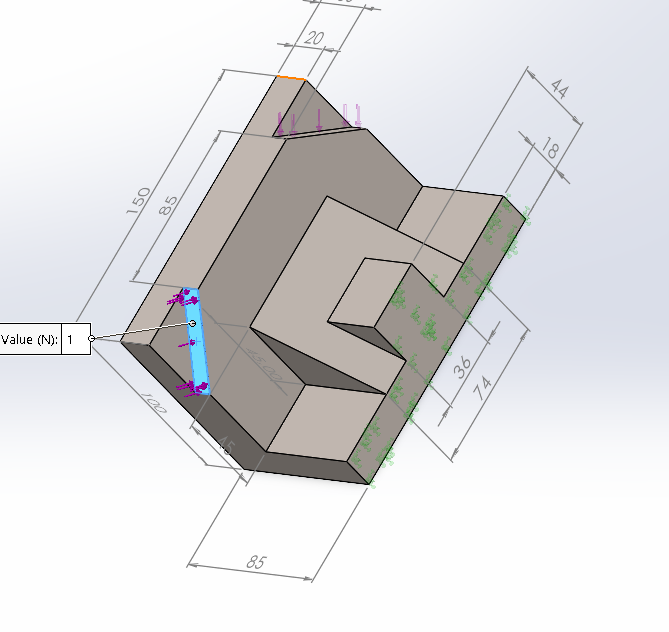
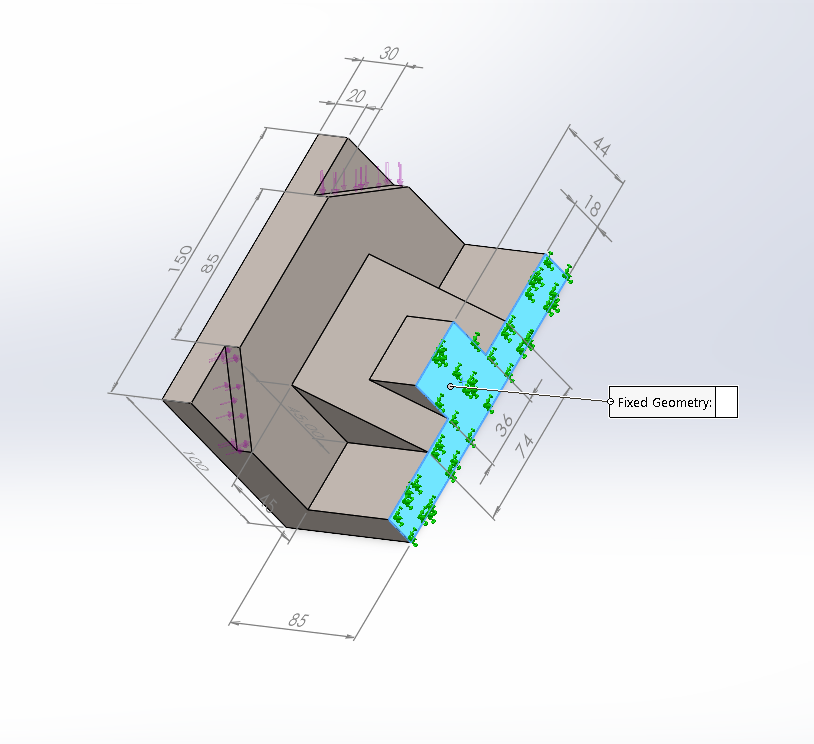
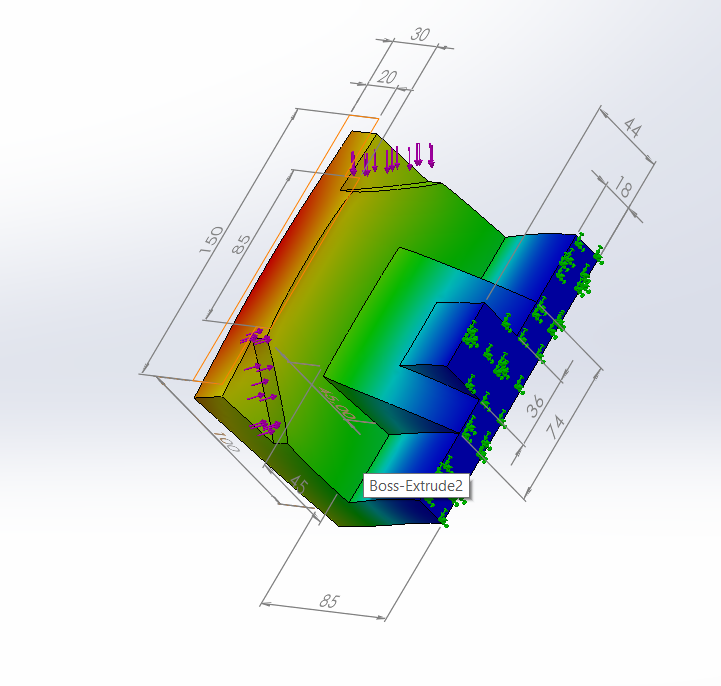
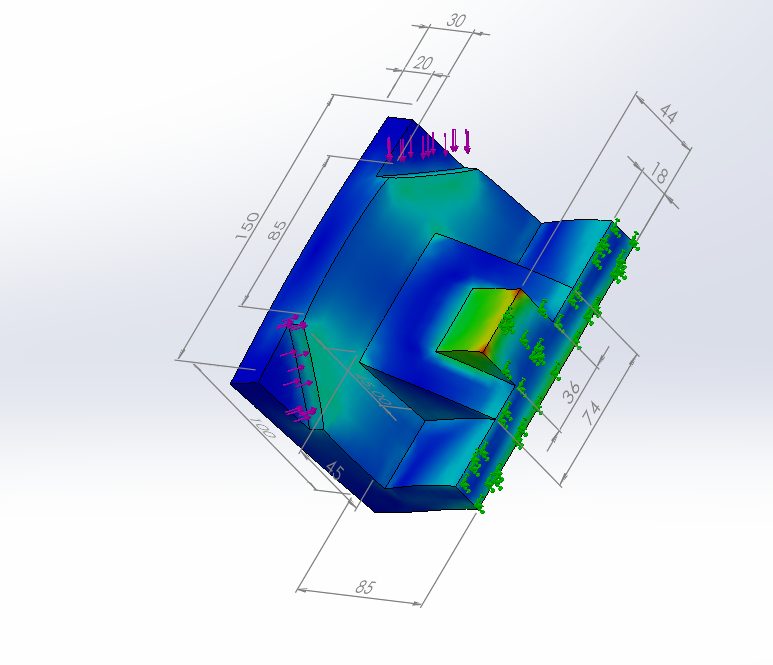
****

Рис.2 – Результаты исследования на напряжение

****

****

1. **Подключение компонента SolidWorks Simulation**
2. **Создание экземпляра статического исследования**
3. **Определение для модели детали материала**
4. **Определение ограничений (закрепление граней, рёбер и др.)**
5. **Определение внешних нагрузок**
6. **Создание сетки модели**
7. **Запуск расчёта исследования**
8. **Просмотр результатов исследования и создание отчёта**

**Результаты исследования напряжения:**

**Визуализация результатов перемещений:**

**Визуализация результатов деформаций:**

**Создание отчёта**

**Вывод:** Получила навыки в процессе проведения инженерного анализа 3D-модели с помощью SW API.

ЛИСТИНГ ПРОГРАММЫ

using SolidWorks.Interop.sldworks;

using SolidWorks.Interop.swconst;

using SolidWorks.Interop.cosworks;

using System;

using System.Runtime.InteropServices;

using System.Windows.Forms;

using System.Collections.Generic;

namespace Lab7

{

public partial class Form1 : Form

{

ISldWorks app = null;

ModelDoc2 doc;

CWModelDoc ActDoc;

private CWStudy Study;

public CWStudyManager StudyMngr;

private CWStaticStudyOptions StatStOpt;

private CWSolidManager SolidMngr;

private int compCount;

public CWSolidComponent SolidComponent { get; private set; }

private string SName;

private PartDoc swPart;

private Body2 swBody;

private CWSolidBody SolidBody;

private dynamic obj;

private byte[] PID;

private dynamic SelObj;

object DistanceValues = null;

object ForceValues = null;

public CWForce CWForce;

CWLoadsAndRestraintsManager LBCMgr = default(CWLoadsAndRestraintsManager);

private CWRestraint CWRestraint;

private CWMesh swMesh;

private List<object> nodeList=new List<object>();

private CWResults results;

private object[] stress;

private object[] strain;

private object[] displacement;

public Form1()

{

InitializeComponent();

}

private void btnConnect\_Click(object sender, EventArgs e)

{

//открыть SolidWorks либо получить экземпляр открытого приложения.

try

{

//попытка открыть либо получить открытый через SolidWorks Api

app = new SldWorks();

//задаём размер окна - на весь экран

app.FrameState = (int)swWindowState\_e.swWindowMaximized;

//делаем окно видимым

app.Visible = true;

}

catch

{

try

{

//если перый вариант не сработал - пробуем получить открытый SolidWorks

app = (SldWorks)Marshal.GetActiveObject("SldWorks.Application");

}

catch

{

//если и это не помогло, тогда что-то не так

MessageBox.Show("Не удалось открыть SolidWorks либо найти открытое приложение.");

return;

}

}

doc = app.ActiveDoc;

//открыть деталь

//если нет открытого чертежа

if (app.ActiveDoc == null)

{

//создать и открыть

doc = (ModelDoc2)app.INewPart();

//задать размеры в миллиметрах

doc.SetUnits((short)swLengthUnit\_e.swMM, (short)swFractionDisplay\_e.swDECIMAL, 0, 0, false);

}

label1.Text = "Соединение установлено!";

btnAnalyze.Enabled = true;

}

private void btnAnalyze\_Click(object sender, EventArgs e)

{

dynamic COSMOSWORKS = default(dynamic);

dynamic COSMOSObject = default(dynamic);

int errorCode;

// Determine host SOLIDWORKS major version

int swVersion = Convert.ToInt32(app.RevisionNumber().Substring(0, 2));

// Calculate the version-specific ProgID of the Simulation add-in that is compatible with this version of SOLIDWORKS

int cwVersion = swVersion - 15;

String cwProgID = String.Format("SldWorks.Simulation.{0}", cwVersion);

// Get the SOLIDWORKS Simulation object

COSMOSObject = app.GetAddInObject(cwProgID);

if (COSMOSObject == null)

{

MessageBox.Show("COSMOSObject object not found");

return;

}

COSMOSWORKS = COSMOSObject.CosmosWorks;

if (COSMOSWORKS == null)

{

MessageBox.Show("COSMOSWORKS object not found");

return;

}

ActDoc = (CWModelDoc)COSMOSWORKS.ActiveDoc;

if (ActDoc == null)

{

MessageBox.Show("No active document");

return;

}

StudyMngr = (CWStudyManager)ActDoc.StudyManager;

if (StudyMngr == null)

{

MessageBox.Show("CWStudyManager object not created");

return;

}

int errCode;

Study = (CWStudy)StudyMngr.CreateNewStudy("Статический-1", (int)swsAnalysisStudyType\_e.swsAnalysisStudyTypeStatic, (int)swsMeshType\_e.swsMeshTypeSolid, out errCode);

if (Study == null)

{

MessageBox.Show("Study not created");

return;

}

StatStOpt = Study.StaticStudyOptions;

SolidMngr = (CWSolidManager)Study.SolidManager;

compCount = SolidMngr.ComponentCount;

SolidComponent = SolidMngr.GetComponentAt(0, out errorCode);

if (SolidComponent == null)

{

MessageBox.Show("CWSolidComponent object not created");

return;

}

//создание экземпляра исследования

SName = SolidComponent.ComponentName;

swPart = (PartDoc)doc;

swBody =swPart.GetBodies2((int)swBodyType\_e.swAllBodies, false)[0];

SolidBody = (CWSolidBody)SolidComponent.GetSolidBodyAt(0, out errCode);

//определние материала

int i=SolidBody.SetLibraryMaterial("C:\\Program Files\\SOLIDWORKS Corp\\SOLIDWORKS\\lang\\english\\sldmaterials\\solidworks materials.sldmat", "201 Annealed Stainless Steel (SS)");

//определение места крепления

List<ICWRestraint> fixedFaces = new List<ICWRestraint>();

ICWLoadsAndRestraintsManager restraintsManager = Study.LoadsAndRestraintsManager;

object[] DispArray1 = { GetFaces((ModelDoc2)doc)[4] };

LBCMgr = (CWLoadsAndRestraintsManager)Study.LoadsAndRestraintsManager;

CWRestraint = LBCMgr.AddRestraint((int)swsRestraintType\_e.swsRestraintTypeFixed, (DispArray1), null, out errorCode);

//определение внешней нагрузки (силы)

object[] DispArray2 = { GetFaces((ModelDoc2)doc)[23] };

object[] DispArray3 = { GetFaces((ModelDoc2)doc)[22] };

object[] ComponentValues = { 2.0, 3.0, 1.0, 1.5, 1.0, 1.0 };

CWForce = LBCMgr.AddForce3((int)swsForceType\_e.swsForceTypeNormal, 0, -1, 0, 0, 0, (DistanceValues), (ForceValues), true, false, 0, 0, 0, 1.0, (ComponentValues), false, false, (DispArray2), null, false, out errCode);

var CWForce1 = LBCMgr.AddForce3((int)swsForceType\_e.swsForceTypeNormal, 0, -1, 0, 0, 0, (DistanceValues), (ForceValues), true, false, 0, 0, 0, 1.0, (ComponentValues), false, false, (DispArray3), null, false, out errCode);

if (errCode != 0) MessageBox.Show("No force applied");

//создание сетки модели

double el, tl;

swMesh = Study.Mesh;

swMesh.GetDefaultElementSizeAndTolerance(0, out el, out tl);

errCode = Study.CreateMesh(0, el, tl);

if (errCode != 0)

MessageBox.Show("Mesh failed");

swMesh = Study.Mesh;

nodeList.Add(swMesh.GetNodes());

StudyMngr.ActiveStudy = 0;

Study = StudyMngr.GetStudy(0);

errorCode = Study.RunAnalysis();

if (errorCode != 0) { MessageBox.Show("Analysis failed"); return; }

results = Study.Results;

//получение результатов по напряжению

stress = (object[])results.GetStress(0, 0, null, (int)swsStrengthUnit\_e.swsStrengthUnitNewtonPerSquareMillimeter, out errorCode);

for (i = stress.GetLowerBound(0); i <= stress.GetUpperBound(0); i += 12)

{

dgvStress.Rows.Add(stress[i], stress[i + 1], stress[i + 2], stress[i + 3], stress[i + 4], stress[i + 5], stress[i + 6], stress[i + 7], stress[i + 8], stress[i + 9], stress[i + 10], stress[i + 11]);

}

//получение результатов по деформации

strain = (object[])results.GetStrain(0, 0, null, out errorCode);

for (i = strain.GetLowerBound(0); i <= stress.GetUpperBound(0); i += 13)

{

dgvStrain.Rows.Add(strain[i], strain[i + 1], strain[i + 2], strain[i + 3], strain[i + 4], strain[i + 5], strain[i + 6], strain[i + 7], strain[i + 8]);

}

//получение результатов по перемещению

displacement = (object[])results.GetTranslationalDisplacement(0, null, 3, out errorCode);

for (i = displacement.GetLowerBound(0); i <= displacement.GetUpperBound(0); i += 5)

{

dgvDisplacement.Rows.Add(displacement[i], displacement[i + 4]);

}

MessageBox.Show("Результаты исследования получены");

}

private static object[] GetFaces(ModelDoc2 swDoc)

{

HashSet<object> result = new HashSet<object>();

object[] features = swDoc.FeatureManager.GetFeatures(true);

object[] faces1 = new object[100];

int i = 0;

foreach (object feature in features)

{

object[] faces = (feature as Feature).GetFaces();

if (faces != null)

{

foreach (object face in faces)

{

result.Add(face);

faces1[i] = face;

i++;

}

}

}

return faces1;

//return new HashSet<Face>(result.Cast<Face>());

}

}

}