SteelBlaze Steel Design Library

Quick Guide

Computations & Graphics, Inc.

Highlands Ranch, CO 80130, USA
Email: info@cg-inc.com
Web: www.cg-inc.com

Introduction

Computations & Graphics, Inc. (CGI) SteelBlaze Steel Design Library is a powerful steel section calculation API (Application Programming Interface) according to AISC 360-22, AISC 360-16, and AISC 360-10. It is based on the solver engine in our sCheck steel section check and design software . You can use this reliable and user-friendly API to develop your custom software royalty-free.

The following are the main features:

- Check the capacity of any of the standard AISC shapes (W, M, S, HP, C, MC, L, WT, MT, ST, 2L, HSS, PIPE) against a set of load effects
- Design and select optimal standard AISC shapes against a set of load effects.
- Consider moment magnification for non-sway condition.
- Auto generate detailed calculation procedures in Word and PDF formats. **Please note that a proper Aspose.Words for C++ license is required to use this report feature. For more information, visit Aspose's website at https://aspose.com.**
- Easy-to-use programming interfaces to define input, solve, and retrieve section results.
- Support various American Institute of Steel Construction codes including AISC 360-22, AISC 360-16, and AISC 360-10.
- Support two-way communication with sCheck through file save and file open.
- Support .NET 4.0, 4.5x, 4.6x, 4.7x, 4.8 and .NET Core 5.0, 6.0, 7.0 in C# and VB.NET languages.
- Support native C++ language.
- Support both 64-bit and 32-bit CPUs architecture on Windows.
- Support Unicode and Non-Unicode in Microsoft Visual Studio 2015 or above.
- Available in both binary library and source code forms.
- A free copy of Professional sCheck Program.
- Numerous source code samples in C++ and C# are freely available.
- Technical support by the SteelBlaze author.
- Reasonably priced and royalty-free.

Sample code included in the install can be readily compiled and run using Visual Studio 2022. You should be able to open the sample code in Visual Studio 2015 and above. For distribution, you may need to install Visual C++ 2015~2022 x86 or x64 redistributables as the SteelBlaze library links dynamically with the visual C++ runtime library.

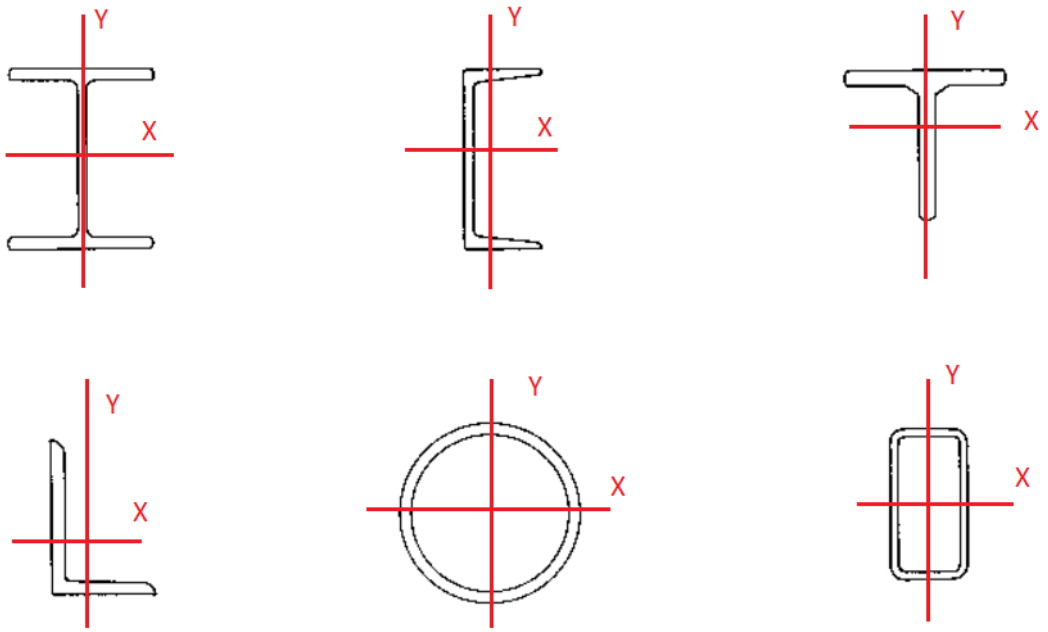
Since SteelBlaze is based on the steel software sCheck, it is highly recommended that you install sCheck and get familiar with its technical backgrounds and capabilities. You can download an evaluation of sCheck from http://www.cg-inc.com/download/download.

The following image shows the file structure of the SDK. It includes a **Bin** folder which contains executables for both native DLLs, .Net and .Net Core assemblies for different CPUs platform and character sets, an **Include** folder which contains C++ header files, a **Lib** folder which contains library files for C++ linking, a **SampleCode** folder which contains example C++ and .Net code for Visual Studio 2022.

The file cgiSteelBlaze.dll is a Windows native DLL. Make sure these files have consistent CPU architecture and are present in the executable directory.

Section Orientations

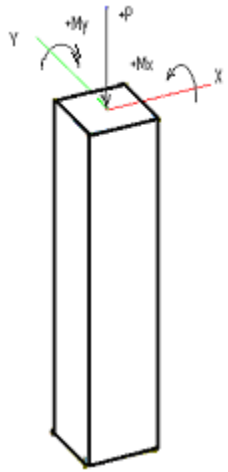
The orientations of section local X and Y axes of various AISC shapes are shown below.



Member Internal Forces and Moments

1. Axial force P acts perpendicular to the section. Moments M_x and M_y act about section local X and Y axes respectively. They have the following sign conventions.

Axial Force P: positive for compression; negative for tension
Moment M_x: Positive when section top most fiber is under compression.
Moment M_y: Positive when section rightmost fiber is under compression.



2. All moments are referenced about the geometric centroid of the gross section.

3. Unlimited number of factored load sets can be input. Loads are the required strength computed by the code-specified factored load combinations using either hands or analysis program such as Real3D-Analysis. It is assumed that an overall 2nd order P-Delta (P-Δ) analysis has been performed on a sway structure. If desired, the software uses moment magnification procedure to calculate the P-delta (P-δ) effect, which accounts for slenderness of columns in non-sway structure or for slenderness along the lengths of columns in sway structure.

4. Critical ratio is computed for each section based on the magnified factored loads and the capacity of the section. Critical ratio equal or less than 1.0 means the design strength is greater than the required strength and the section is adequate. Critical ratio greater than 1.0 means the design strength is less than the required strength and the section is inadequate.

Section Check

The following are inputs and results exposed through the API when checking a single AISC section.

Lux, Luy and Luz are unbraced lengths in feet in local x, y, and z directions. Kx, Ky, and Kz are unbraced length factors in local x, y and z directions.

Lb is the unbraced lateral length in feet. Cb is the lateral-torsional buckling modification factor for non-uniform moment diagrams. It should be greater or equal to 1.0. You can use 1.0 for Cb conservatively.

Connector Distance is used for double angles only.

Pu, Mux, Muy, Vux, Vuy are required axial force, major moment, minor moment, major shear and minor shear. For Pu, the compressive force is positive while tensile force is negative. Moment Mux is positive when section top most fiber is under compression. Moment Muy is positive when section rightmost fiber is under compression. Moment magnification may be optionally considered to account for the P-delta (P-δ) effect. All forces are in kips and all moments are in ft-kips.

If direct analysis method is chosen, the program will account for stiffness reduction when calculating the moment magnification factor B_{1x} and B_{1y} for P- δ effects. It is assumed that a P- Δ (only) second-order analysis is performed for the load effects (P_u, M_{ux}, M_{uy}, V_{ux}, and V_{uy}).

Cmx, Cmy are coefficients accounting for non-uniform moments when computing moment magnification. You can use 1.0 for Cmx and Cmy conservatively. If 0 is entered for Cmx or Cmy, 1.0 is used in the computation instead.

The actual section check is achieved by calling the interface function solve().

Results include axial capacity (phi-Pn), moment capacity (phi-Mnx, phi-Mny), shear capacity (phi-Vnx, phi-Vny), moment magnification factors (B1x, B1y), and critical ratio. The section is deemed safe to resist a load if the critical ratio is less than 1.0, otherwise, the section is deemed unsafe. Please note that for a single angle, the moment capacities are given about the principal w-w and z-z axes and the input moments Mux and Muy are transformed in the principal axes before flexural-axial interaction ratio is checked.

Section Design

The section design process generally involves trying and checking multiple sections, and select the first few candidate sections that have critical ratios less than 1.0. The trial sections can be generated based on either the section prefixes (e.g. W12, W14) or the section dimension limits. The API provides couple of generateSections() interface functions to generate the trial sections. *The actual section design is achieved through calling the interface function solve_multiple().*

AISC Section Database

The AISC section database used in SteelBlaze is based on AISC steel shapes. For Unicode configuration, the section database contains the section index text file aisc16.idx and section property values binary file aisc16.tbl. For Non-Unicode configuration, the section database contains the section index text file aisc16.idx and section property values binary file aisc16_a.tbl. The section index text and property values binary files must be placed under the same directory as the main executables.

You can customize the section database by exporting the current section database to a comma-separated values (csv) file, adding/deleting/updating sections and their properties in the file, and then importing the modified csv file. The relevant interface functions are exportSectionDatabase() and importSectionDatabase().

The section properties comprise 40 double-precision values, each occupying 8 bytes, followed by an English section label consisting of either 40 characters encoded in 2 bytes each (Unicode) or 40 characters encoded in 1 byte each (Non-Unicode). This is succeeded by a metric section label, which also consists of either 40 characters encoded in 2 bytes each (Unicode) or 40 characters encoded in 1 byte each (Non-Unicode). All measurements for section value properties are based in inches.

Report Feature

SteelBlaze offers a very nice report feature: the ability to generate step-by-step calculation procedures in calculating section axial, flexural, and shear capacities, and critical ratios against a set of loads. This report feature requires Aspose.Words for C++ library. Therefore, you need to purchase a proper Aspose.Words for C++ license in order to use the report feature in SteelBlaze. For more information, visit Aspose's website at https://aspose.com. The license file Aspose.Words.CPP.lic must be placed under the same directory as the main executables.

Communication with sCheck

SteelBlaze can save data in sCheck file format (.sck) and open an existing sCheck file.

C++ Interface

The C++ library contains both a 64-bit and 32-bit Windows DLL cgiSteelBlaze.dll. The interface includes the following header files:_cgiDefines.h, _cgiISteelStructure.h, _cgiPlatform.h and cgiSteelBlaze.h. It also includes a cgiSteelBlaze.lib for linking to your projects.

The _cgiDefines.h defines all input and output data structures. _cgiISteelStructure.h is the one and only interface to set input, perform analysis and design, and retrieve output. The best way to learn how you use these data structures and interfaces is to study the examples included in the C++ console application project cgiSteelBlazeClient.

The following lists all the interface functions exposed by cgiISteelStructure:

```
// function callbacks to notify the clients
typedef void (*fnLISTMSG)(LPCTSTR sz0, LPCTSTR sz1);
typedef void (*fnSTATUSMSG)(LPCTSTR sz);

struct CGISTEELBLAZE_API cgiISteelStructure
{
        virtual void setListMessageFunction(fnLISTMSG fnListMsg) = 0;
        virtual void setStatusMessageFunction(fnSTATUSMSG fnStatusMsg) = 0;

        virtual void getExePath(TCHAR exePath[], int size)const = 0;
        virtual int getLastError()const = 0;
        virtual void clearLastError() = 0;
        virtual void setShowSolverMessageBox(bool bShow) = 0;
        virtual bool getShowSolverMessageBox()const = 0;
        virtual void setAbortSolutionKey(int key) = 0;
        virtual int getAbortSolutionKey()const = 0;

        virtual bool setSteelCode(int code) = 0;
        virtual int getSteelCode()const = 0;
        virtual int getAiscSection(cgiSteelBlazeNamespace::cgiAiscSection& section, const TCHAR sectionLabel[])const = 0;  // return -1 if not found
        virtual void setPrint(bool print) = 0;
        virtual bool getPrint()const = 0;
        virtual void setMaterial(double fy) = 0;         // fy in ksi
        virtual double getMaterial()const = 0;
        virtual void setUseDirectDesign(bool use) = 0;
        virtual bool getUseDirectDesign()const = 0;
        virtual void setConsiderMomentMagnification(bool consider) = 0;
        virtual bool getConsiderMomentMagnification()const = 0;
        virtual void setLateralTorsonalProperties(double Cb, double Lb) = 0;      // Lb in feet
        virtual void getLateralTorsonalProperties(double& Cb, double& Lb)const = 0;       // Lb in feet
        virtual void setDoubleAngleConnectorDistance(double distance) = 0;        // distance in feet
        virtual double getDoubleAngleConnectorDistance() = 0;     // in feet
        virtual void setColumnSlenderness(double Kx, double Ky, double Kz, double Lux, double Luy, double Luz) = 0;       // Lux, Luy, and Luz in feet
```

```
        virtual void getColumnSlenderness(double& Kx, double& Ky, double& Kz, double& Lux, double& Luy, double& Luz) const = 0;    // Lux, Luy, and Luz in
feet

        virtual bool addSectionLoad(const cgiSteelBlazeNamespace::cgiSectionLoad& sectionLoad) = 0;
        virtual void clearSectionLoads() = 0;

        virtual bool generateSections(cgiSteelBlazeNamespace::cgiAiscSection*& pSections, int& sectionCount, int iShape, const TCHAR sectionPrefix[]) = 0;
        // minWidth, maxWidth, minDepth, maxDepth in inches
        virtual bool generateSections(cgiSteelBlazeNamespace::cgiAiscSection*& pSections, int& sectionCount, int iShape, double minWidth, double maxWidth,
                                      double minDepth, double maxDepth) = 0;
        virtual bool exportSectionDatabase(const TCHAR csvFile[]) const = 0;
        virtual bool importSectionDatabase(const TCHAR csvFile[]) const = 0;

        virtual bool checkInput()const = 0;
        virtual bool solve(const TCHAR sectionLabel[], cgiSteelBlazeNamespace::cgiSectionResult*& pResult, int& resultCount,
                           const TCHAR reportPath[] = nullptr, bool showReport = false) = 0; // resultCount equals load count
        virtual bool solve_multiple(const cgiSteelBlazeNamespace::cgiAiscSection* pSections, int sectionCount, int maximumCandidates,
                                    cgiSteelBlazeNamespace::cgiDesignSection*& pDesignSections, int& designSectionCount) = 0;
        virtual bool solve_multiple(int iShape, const TCHAR sectionPrefix[], int maximumCandidates,
                                    cgiSteelBlazeNamespace::cgiDesignSection*& pDesignSections, int& designSectionCount) = 0;
        virtual bool solve_multiple(const int iShape, double minWidth, double maxWidth, double minDepth, double maxDepth, int maximumCandidates,
                                    cgiSteelBlazeNamespace::cgiDesignSection*& pDesignSections, int& designSectionCount) = 0;
        virtual bool save(const TCHAR fileName[]) const = 0;
        virtual bool open(const TCHAR fileName[]) = 0;

        virtual void deleteMemoryArray(cgiSteelBlazeNamespace::cgiAiscSection* p)const = 0;
        virtual void deleteMemoryArray(cgiSteelBlazeNamespace::cgiSectionResult* p)const = 0;
        virtual void deleteMemoryArray(cgiSteelBlazeNamespace::cgiDesignSection* p)const = 0;
};

CGISTEELBLAZE_API cgiISteelStructure* CreateSteelStructure();
CGISTEELBLAZE_API void DeleteSteelStructure(cgiISteelStructure* pStructure);
```

The following lists a few common enums in namespace cgiSteelBlazeNamespace.

```
enum { kAISC_14_LRFD, kAISC_15_LRFD, kAISC_16_LRFD, kCode_End };

enum {
        kErrorNone, kInvalidInput, kInvalidDesignCode, kSolverError, kAbnormalSolverTermination, kReportError, kFileAccessError, kLicenseError,
        kNoResultAvailable, kInvalidSectionType, kInvalidSection, kUnknownError
};

enum {
        AISC_W, AISC_M, AISC_S, AISC_HP, AISC_C, AISC_MC,
        AISC_L, AISC_WT, AISC_MT, AISC_ST, AISC_2L, AISC_HSS, AISC_HSSP, AISC_PIPE, AISC_TS, AISC_SHAPE_COUNT
};

enum {
        T_F, W, A, d, Ht, OD, bf, b, ID, tw, tf, t, tnom, tdes,
        kdes, kdet, x, y, e0, xp, yp, bf_2tf, b_t, h_tw, h_tdes, D_t, // kdtl=kdet, h_t=h_tdes
        Fy3p, X1, X2, Ix, Zx, Sx, rx, Iy, Zy, Sy, ry, rz, J, Cw, C,
        Wno, Sw1, Qf, Qw, r0, H, tan_alpha, Qs, // Sw=Sw1

        Sw2, Sw3, Iw, SwA, SwB, SwC, SzA, SzB, SzC, rts, h0, PA, PB,
        Iz, Sz,
        ddet, h, bfdet, B, twdet, twdet_2, tfdet, k1,
        b_tdes, zA, zB, zC, wA, wB, wC,
        PA2, PC, PD, T, WGi, WG0,
        temp_18, temp_19, temp_20,
        temp_21, temp_22, temp_23, temp_24, temp_25, temp_26, temp_27, temp_28, temp_29, temp_30,
        PROP_END
};
```

The following lists a few result data structures in namespace cgiSteelBlazeNamespace

```
struct CGISTEELBLAZE_API cgiSectionLoad
{
        int iId;        // load id
        double fPu;     // in kips
        double fMux;    // in ft-kips
        double fMuy;    // in ft-kips
        double fCmx;
        double fCmy;
        double fVux;    // in kips
        double fVuy;    // in kips

};

struct CGISTEELBLAZE_API cgiSectionResult
{
        int iId;                // load id
        double phiPn;           // in kips
        double phiMnx;          // in ft-kips
```

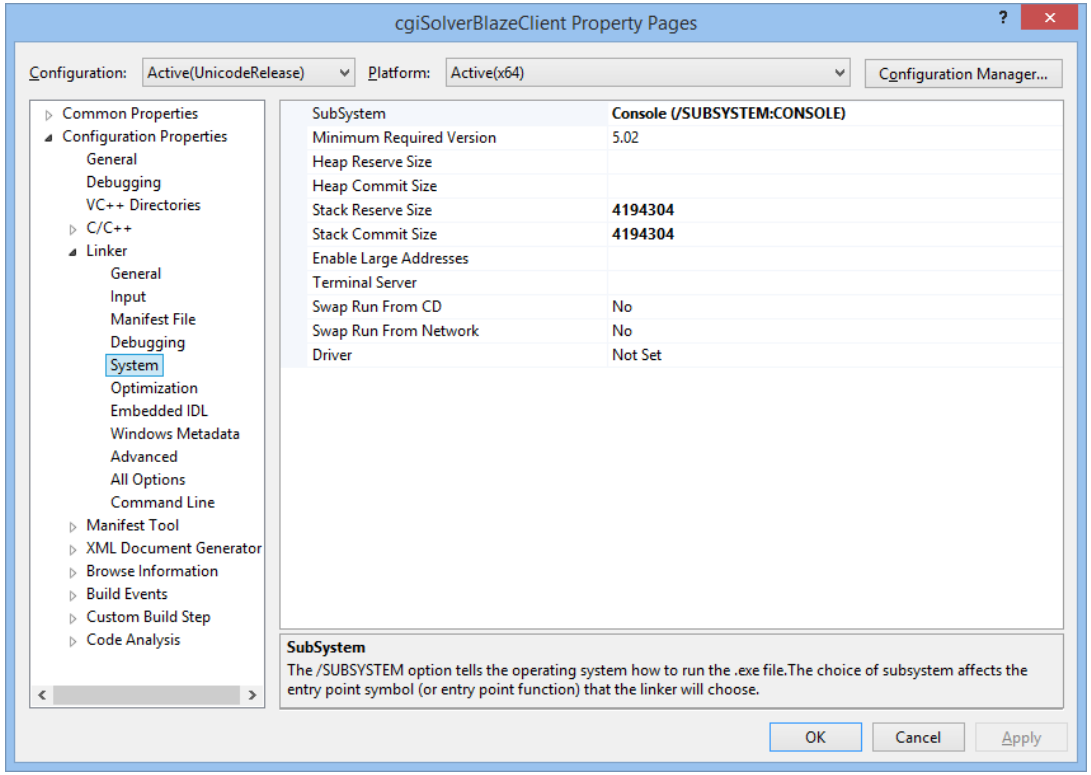
```
        double phiMny;          // in ft-kips
        double phiVnx;          // in kips
        double phiVny;          // in kips
        double B1x;
        double B1y;
        double criticalRatio;
};

// cgiAiscSection must sync with jxSect_AISC
struct CGISTEELBLAZE_API cgiAiscSection
{
        double fVal[PROP_END];   // inch based
        TCHAR szLabel_E[40];    // AISC_Manual_Label
        TCHAR szLabel_M[40];    // EDI_Std_Nomenclature
};

struct CGISTEELBLAZE_API cgiDesignSection
{
        double criticalRatio;
        int loadIndex;
        TCHAR szSectionLabel[40];

};
```

Four versions of cgiSteelBlaze.lib and cgiSteelBlaze.dll are provided for your configuration needs: Unicode or Multi-Bytes, 86x or 64x CPUs. You should link your projects to the correct version of the lib file. Make sure you also copy the correct version of the DLLs (cgiSteelBlaze.dll) to your executable directory. If your project is configured for 64-bit CPU, you may need to increase the stack reserve size and stack commit size from the default 1 MB to something larger (say 4 MB) as shown below.


cgiSolverBlazeClient Property Pages
Configuration: Active(UnicodeRelease)
Platform: Active(x64)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
SubSystem
Console (/SUBSYSTEM:CONSOLE)
Minimum Required Version
5.02
Heap Reserve Size
Heap Commit Size
Stack Reserve Size
4194304
Stack Commit Size
4194304
Enable Large Addresses
Terminal Server
Swap Run From CD
No
Swap Run From Network
No
Driver
Not Set
SubSystem
The /SUBSYSTEM option tells the operating system how to run the .exe file.The choice of subsystem affects the entry point symbol (or entry point function) that the linker will choose.
OK
Cancel
Apply

Example

The following is an example of using SteelBlaze to investigate an AISC section W10x33 under a set of loads

```
#include "_cgiISteelStructure.h"
#include <iostream>
#include <iomanip>
#include <vector>
#include <fstream>
#include <iostream>
#include <sstream>

using namespace std;
using namespace cgiSteelBlazeNamespace;

#ifdef _UNICODE
#define COUT wcout
#else
#define COUT cout
#endif

static void ListMsg(LPCTSTR sz0, LPCTSTR sz1)
{
    COUT << sz0 << "\t" << sz1 << endl;
}

static void StatusMsg(LPCTSTR sz)
{
    COUT << "STATUS ______ " << sz << endl;
}

bool equal_for_double(double d1, double d2) {
    return fabs(d1 - d2) <= 1e-6;
}

void verify_example_singleSection()
{
    // create a structural model
    cgiISteelStructure* pStructure = CreateSteelStructure();

    //auto export1 = pStructure->exportSectionDatabase(_T("c:\\temp\\aa.csv"));
    //auto import = pStructure->importSectionDatabase(_T("c:\\temp\\aa.csv"));

    // message functions, can be set null in which case no messages will be printed during solution
    pStructure->setListMessageFunction(ListMsg);
    pStructure->setStatusMessageFunction(StatusMsg);

    TCHAR exePath[_MAX_PATH];
    pStructure->getExePath(exePath, _MAX_PATH - 1);
```

```
    COUT << _T("exePath = ") << exePath << endl;

    pStructure->setPrint(true);
    pStructure->setUseDirectDesign(false);
    pStructure->setConsiderMomentMagnification(true);
    pStructure->setMaterial(50);
    pStructure->setColumnSlenderness(1.0, 1.0, 1.0, 14.0, 14.0, 14.0);
    pStructure->setLateralTorsonalProperties(1.14, 14.0);
    pStructure->setDoubleAngleConnectorDistance(0);

    cgiSectionLoad load;
    load.fPu = 30;
    load.fMux = 90;
    load.fMuy = 12;
    load.fVux = 0;
    load.fVuy = 0;
    load.fCmx = 1;
    load.fCmy = 1;
    pStructure->addSectionLoad(load);

    cgiSteelBlazeNamespace::cgiSectionResult* pResult = nullptr;
    int resultCount = 0;
    bool success = pStructure->solve(_T("W10x33"), pResult, resultCount, _T("C:\\temp\\1.pdf"), true);
    //bool success = pStructure->solve(_T("W10x33"), pResult, resultCount);
    if (!success) {
        COUT << "error with code = " << pStructure->getLastError() << endl;
        return;
    }
    for (int i = 0; i < resultCount; i++)
    {
        const auto& result = pResult[i];
        COUT << "iId=" << result.iId << ", phi-Pnx=" << result.phiPn << ", phi-Mnx=" << result.phiMnx << ", phi-Mny=" << result.phiMny
            << ", phi-Vnx=" << result.phiVnx << ", phi-Vny=" << result.phiVny << ", B1x=" << result.B1x << ", B1y=" << result.B1y
            << ", critical ratio=" << result.criticalRatio << endl;
    }
    pStructure->deleteMemoryArray(pResult);

    DeleteSteelStructure(pStructure);

}
```

.NET Interface

The .NET interface is a class library assembly cgiSteelBlazeCli.dll. It is written in C++/CLI and forwards calls to a native Windows DLL called cgiSteelBlaze.dll. The actual interface functions are contained in a single class called `cgiSteelBlazeCli` as listed in the following. The .NET Core interface is a class library assembly cgiSteelBlazeCoreCli.dll. The interface functions are identical to those in .NET Framework interface.

Different versions of cgiSteelBlazeCli.dll are provided to work with .NET Framework 4.0, 4.5x, 4.6x, 4.7x, and 4.8x on x86 and x64 CPUs. Different versions of cgiSteelBlazeCoreCli.dll are provided to work with .NET Core 5.0, 6.0, 7.0 on x86 and x64 CPUs. Make sure your project has reference to the correct version of cgiSteelBlazeCli.dll or cgiSteelBlazeCoreCli.dll. If your project is configured to build for both x86 and x64 CPUs, make sure to set "Copy Local" to false for the DLL reference and manually copy the correct version of cgiSteelBlazeCli.dll or cgiSteelBlazeCoreCli.dll to the executable directory. Also make sure a copy of the native dependent DLLs (cgiSteelBlaze.dll) are placed in the executable directory. For .NET Core, ijwhost.dll is a shim for finding and loading the runtime. All C++/CLI libraries are linked to this shim, such that ijwhost.dll is found/loaded when the C++/CLI library is loaded. Therefore, make sure a copy of ijwhost.dll (e.g.: C:\Program Files\dotnet\packs\Microsoft.NETCore.App.Host.win-x64\7.0.0\runtimes\win-x64\native\Ijwhost.dll) is placed in the executable directory as well.

`cgiSteelBlazeCli` is the one and only interface to set input, perform analysis and design, and retrieve output. The best way to learn how you use these data structures and interfaces is to study the examples included in the C# console application project cgiSteelBlazeTestCSharp.

The following lists all the interface functions exposed by cgiSteelBlazeClass in cgiSteelBlazeCli namespace for .NET and cgiSteelBlazeCoreCli namespace for .NET Core:

```
namespace cgiSteelBlazeCli
{
    public class cgiSteelBlazeClass : IDisposable
    {
        public void setListMessageFunction(cgiSteelBlazeClass.ListMessageDelegate fnListMsg);
        public void setStatusMessageFunction(cgiSteelBlazeClass.StatusMessageDelegate fnStatusMsg);

        public bool createStructure();
        public void getExePath(ref StringBuilder exePath);
        public cgiErrorEnum getLastError();
        public void clearLastError();
        public void setShowSolverMessageBox(bool bShow);
        public bool getShowSolverMessageBox();
        public void setAbortSolutionKey(int key);
        public int getAbortSolutionKey();

        public bool setSteelCode(cgiSteelCodeEnum code);
```

```
        public cgiSteelCodeEnum getSteelCode();
        public int getAiscSection(ref cgiAiscSectionCli section, string sectionLabel);
        public void setPrint(bool print);
        public bool getPrint();
        public void setMaterial(double fy);              // fy in ksi
        public double getMaterial();
        public void setUseDirectDesign(bool use);
        public bool getUseDirectDesign();
        public void setConsiderMomentMagnification(bool consider);
        public bool getConsiderMomentMagnification();
        public void setLength(double length);
        public double getLength();
        public void setLateralTorsonalProperties(double Cb, double Lb);           // Lb in ft
        public void getLateralTorsonalProperties(ref double Cb, ref double Lb);   // Lb in ft
        public void setColumnSlenderness(double Kx, double Ky, double Kz, double Lux, double Luy, double Luz);     // Lux, Luy, Luz in ft
        public void getColumnSlenderness(ref double Kx, ref double Ky, ref double Kz, ref double Lux, ref double Luy, ref double Luz);     // Lux, Luy, Luz
in ft
        public void setDoubleAngleConnectorDistance(double distance);     // distance in ft
        public double getDoubleAngleConnectorDistance(); // in ft

        public bool addSectionLoad(cgiSectionLoadCli sectionLoad);
        public void clearSectionLoads();

        public bool generateSections(ref List<cgiAiscSectionCli> pSections, cgiSectionType iShape,
                                     double minWidth, double maxWidth, double minDepth, double maxDepth);
        public bool generateSections(ref List<cgiAiscSectionCli> pSections, cgiSectionType iShape, string sectionPrefix);

        public bool exportSectionDatabase(string csvFile);
        public bool importSectionDatabase(string csvFile);

        public bool checkInput();
        public bool solve(string sectionLabel, ref List<cgiSectionResultCli> sectionResults, string reportPath, bool showReport);
        public bool solve_multiple(List<cgiAiscSectionCli> pSections, int maximumCandidates, ref List<cgiDesignSectionCli> pDesignSections);
        public bool solve_multiple(cgiSectionType iShape, string sectionPrefix, int maximumCandidates, ref List<cgiDesignSectionCli> pDesignSections);
        public bool solve_multiple(cgiSectionType iShape, double minWidth, double maxWidth, double minDepth, double maxDepth,
                                   int maximumCandidates, ref List<cgiDesignSectionCli> pDesignSections);
        public bool save(string filePath);
        public bool open(string filePath);


        [UnmanagedFunctionPointer(CallingConvention.Cdecl, CharSet = CharSet.Unicode)]
        public delegate void ListMessageDelegate(string A_0, string A_1);

        [UnmanagedFunctionPointer(CallingConvention.Cdecl, CharSet = CharSet.Unicode)]
        public delegate void StatusMessageDelegate(string A_0);
    }
}
```

The following lists a few useful enums.

```
namespace cgiSteelBlazeCli
```

```
{
    public enum cgiSteelCodeEnum
    {
        kAISC_14_LRFD,
        kAISC_15_LRFD,
        kAISC_16_LRFD,
        kCode_End,
    }

    public enum cgiErrorEnum
    {
        kErrorNone,
        kInvalidInput,
        kInvalidDesignCode,
        kSolverError,
        kAbnormalSolverTermination,
        kReportError,
        kFileAccessError,
        kLicenseError,
        kNoResultAvailable,
        kInvalidSectionType,
        kInvalidSection,
        kUnknownError,
    }

    public enum cgiSectionType
    {
        AISC_W,
        AISC_M,
        AISC_S,
        AISC_HP,
        AISC_C,
        AISC_MC,
        AISC_L,
        AISC_WT,
        AISC_MT,
        AISC_ST,
        AISC_2L,
        AISC_HSS,
        AISC_HSSP,
        AISC_PIPE,
        AISC_TS,
        AISC_SHAPE_COUNT,
    }

    public enum cgiSectionProperty
    {
        T_F,
        W,
        A,
        d,
        Ht,
```

```
OD,
bf,
b,
ID,
tw,
tf,
t,
tnom,
tdes,
kdes,
kdet,
x,
y,
e0,
xp,
yp,
bf_2tf,
b_t,
h_tw,
h_tdes,
D_t,
Fy3p,
X1,
X2,
Ix,
Zx,
Sx,
rx,
Iy,
Zy,
Sy,
ry,
rz,
J,
Cw,
C,
Wno,
Sw1,
Qf,
Qw,
r0,
H,
tan_alpha,
Qs,
Sw2,
Sw3,
Iw,
SwA,
SwB,
SwC,
SzA,
SzB,
```

```
        SzC,
        rts,
        h0,
        PA,
        PB,
        Iz,
        Sz,
        ddet,
        h,
        bfdet,
        B,
        twdet,
        twdet_2,
        tfdet,
        k1,
        b_tdes,
        zA,
        zB,
        zC,
        wA,
        wB,
        wC,
        PA2,
        PC,
        PD,
        T,
        WGi,
        WG0,
        temp_18,
        temp_19,
        temp_20,
        temp_21,
        temp_22,
        temp_23,
        temp_24,
        temp_25,
        temp_26,
        temp_27,
        temp_28,
        temp_29,
        temp_30,
        PROP_END,
    }
}
```

The following lists result data structures.

```
namespace cgiSteelBlazeCli
{
    public class cgiAiscSectionCli
    {
```

```
        public List<double> fVal;
        public string szLabel_E;
        public string szLabel_M;
    }

    public class cgiDesignSectionCli
    {
        public double criticalRatio;
        public int loadIndex;
        public string szSectionLabel;
    }

    public class cgiSectionLoadCli
    {
        public int iId;
        public double fPu;      // kips
        public double fMux;     // ft-kips
        public double fMuy;     // ft-kips
        public double fCmx;
        public double fCmy;
        public double fVux;     // kips
        public double fVuy;     // kips
    }

    public class cgiSectionResultCli
    {
        public int iId;
        public double phiPn;            // kips
        public double phiMnx;           // ft-kips
        public double phiMny;           // ft-kips
        public double phiVnx;           // kips
        public double phiVny;           // kips
        public double B1x;
        public double B1y;
        public double criticalRatio;
    }

}
```

Example

The following is an example of using SteelBlaze to investigate an AISC section W10x33 under a set of loads

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using cgiSteelBlazeCli;


namespace cgiSteelBlazeTestCSharp
{
    public class Example_SingleSection
    {
        static void Callback(string s0, string s)
        {
            Console.WriteLine("{0}, {1}", s0, s);
        }

        static void StatusCallback(string s)
        {
            Console.WriteLine("{0}", s);
        }

        static public void verify()
        {
            RunModel();
        }

        static bool equal_for_double(double d1, double d2)
        {
            return Math.Abs(d1 - d2) <= 1e-6;
        }

        static public void RunModel()
        {
            cgiSteelBlazeClass solver = new cgiSteelBlazeClass();
            solver.createStructure();

            //var import = solver.importSectionDatabase("c:\\temp\\aa.csv");
            //var export = solver.exportSectionDatabase("c:\\temp\\bb.csv");

            Console.WriteLine("-------------------------------------------------------------------------------");
            Console.WriteLine("-------------------------------------------------------------------------------");

            cgiSteelBlazeClass.ListMessageDelegate listMsg = new cgiSteelBlazeClass.ListMessageDelegate(Callback);
```

```
            cgiSteelBlazeClass.StatusMessageDelegate statusMsg = new cgiSteelBlazeClass.StatusMessageDelegate(StatusCallback);
            solver.setListMessageFunction(listMsg);
            solver.setStatusMessageFunction(statusMsg);

            StringBuilder sb = new StringBuilder();
            solver.getExePath(ref sb);

            solver.setPrint(true);
            solver.setUseDirectDesign(false);
            solver.setConsiderMomentMagnification(true);
            solver.setMaterial(50);
            solver.setColumnSlenderness(1.0, 1.0, 1.0, 14.0, 14.0, 14.0);
            solver.setLateralTorsonalProperties(1.14, 14.0);
            solver.setDoubleAngleConnectorDistance(0);

            cgiSteelBlazeCli.cgiSectionLoadCli load = new cgiSectionLoadCli(0);
            load.fPu = 30;
            load.fMux = 90;
            load.fMuy = 12;
            load.fVux = 0;
            load.fVuy = 0;
            load.fCmx = 1;
            load.fCmy = 1;
            solver.addSectionLoad(load);

            cgiAiscSectionCli aiscSection = new cgiAiscSectionCli();
            solver.getAiscSection(ref aiscSection, "W10x33");

            Console.WriteLine("solving...");

            List<cgiSectionResultCli> list = new List<cgiSectionResultCli>();
            bool success = solver.solve("W10x33", ref list, "C:\\temp\\1.pdf", true);
            if (!success)
            {
                Console.WriteLine("error with code = " + solver.getLastError());
                return;
            }
            for (int i = 0; i < list.Count; i++)
            {
                var result = list[i];
                Console.WriteLine("iId=" + result.iId + ", phi-Pnx=" + result.phiPn + ", phi-Mnx=" + result.phiMnx + ", phi-Mny=" + result.phiMny
                    + ", phi-Vnx=" + result.phiVnx + ", phi-Vny=" + result.phiVny + ", B1x=" + result.B1x + ", B1y=" + result.B1y
                    + ", critical ratio=" + result.criticalRatio);
            }

            Console.WriteLine("solved");

        }

    }
}
```

This last page is left blank on purpose