45 Structural Behavior Analysis and Optimization, Integrating MATLAB with Autodesk Robot

Giulia Cerè, Wanqing Zhao, and Yacine Rezgui

Abstract

The concepts of structural behavior analysis and optimization have started to be combined in the latest decades with an increasing trend also due to the need of often meeting performance targets, high structural complexity and costs. Nonetheless, existing approaches tackled this issue mainly in the domain of static calculations or referring to a specific type of optimization (e.g., size, topology or geometry). A new methodology is proposed to systematically perform different types of analysis (e.g., linear and nonlinear), by exploiting the Autodesk Robot Structural Analysis API through MATLAB. This approach involves the adoption of ActiveX technologies for the manipulation of COM (Component Object Model) objects in the MATLAB environment. A real-world example of linear dynamic modal analysis is also presented and a synthetic diagnostic of the structure is conducted based on the displacements resulting from the calculation.

Keywords

Autodesk robot • MATLAB • Structural behavior analysis • Optimization

45.1 Introduction

With an increasing complexity in the design of structural shapes, a higher impact of natural hazards on the building environment [1] and a growing pressure for delivering projects within shorter timing, the software-based structural behavior analysis and optimization problem have been emerging significantly in the engineering domain, particularly in relation to the design phase [2]. The need of structural behavior analysis adopting a software has become widely implemented in worldwide building regulations, such as Eurocodes [3] and FEMA [4]. As a matter of fact, the introduction of software allowed the designer to avoid cumbersome hand-calculations which become extremely complex and difficult to handle especially for nonlinear analysis and when the simultaneous interaction of several materials has to be taken into account into non-static conditions.

As far as the mere structural behavior analysis is concerned, the commercial panorama of available software packages is wide and offers a considerable range of choices for this purpose. Some of these tools mainly focus on specific construction materials, such as ADAPT [5], which has been developed specifically for reinforced concrete (RC) constructions. On the other hand, a vast range of other software adopt a multi-material approach, qualifying as more versatile tools for building design and analysis. Examples of this category are for instance Tekla [6], SCIA, Midas, SAP2000 [7] and Robot Structural

G. Cerè (✉) · W. Zhao · Y. Rezgui
BRE Trust Centre for Sustainable Engineering, Cardiff School of Engineering, Cardiff University, Cardiff, CF243AB, UK
e-mail: cereg@cardiff.ac.uk

W. Zhao
e-mail: zhaow9@cardiff.ac.uk

Y. Rezgui
e-mail: rezguiy@cardiff.ac.uk

I. Mutis and T. Hartmann (eds.), *Advances in Informatics and Computing in Civil and Construction Engineering*,
https://doi.org/10.1007/978-3-030-00220-6_45

Analysis Professional [8]. In particular, SAP2000 is listed amongst the leading software in the field of structural analysis, given its straightforward and flexible functionalities. Although, the use of SAP2000 most of times requires the integration of Excel in order to conduct the structural members' verifications and additional post-analysis calculations.

The choice of Robot Structural Analysis Professional over other software grounds on several factors. In first instance, Robot is part of a suite, which allows a high interconnectivity and compatibility level between the different software. The two-way interconnectivity with Revit consists in a key feature of Robot, allowing a high quality bidirectional export of the files and hence reducing potential adjustments required by the designer [9]. An additional advantage can be found in the high integration level of the most influencing and up to date worldwide building regulations, while the majority of structural software are provided with local standard and hence confining the use of the tool to a limited area.

On the other hand, optimizing a process consists in matching the design objectives (e.g., performance) with the compliance of a series of constraints (e.g., costs, deflection, base shear), by tuning the design variables (e.g., building and structural elements' size, material) [10]. Several tools are available for this purpose, such as iSIGHT, DOT and MATLAB, the latter perhaps consisting in the most diffused being also implemented with a specific optimization toolbox [11]. Previous research addressed the use of MATLAB in combination with other software to simulate the effects of a seismic solicitation. One of these is the PBEE toolbox [12], which draws on linking MATLAB and OpenSees in order to perform seismic analysis on reinforced concrete (RC) frames. This grounds on the notion of plastic hinges development in correspondence of structural nodes as a representation of the ductile abilities of the building, although confining the work to a mere diagnosis of the structure.

An example of structural optimization can be found in the SMART Sizer plug in devised by Buro Happold in order to be coupled to Robot Autodesk Structural Analysis [2]. This tool allows to optimize the size of the structural members grounding on the results of a displacement analysis in order to evenly distribute the stiffness in the entire frame. Despite the novelty of this approach, this methodology addresses purely size optimization and it is not scalable for other structural analysis or optimization problems. To the best of our knowledge, the integration of the Robot API has been mainly exploited with programming languages such as Visual Basic, C ++ or C# [13], whereas the integration with MATLAB has not been explored thoroughly yet, given its renown in providing an accessible numerical computing environment.

As far as the optimizing strategies are concerned, most of the existing literature broadly identifies three categories, mainly operating on size, geometry (i.e., shape) or topology of the structural members [14, 15]. It has to be pointed out that the majority of the literature does not rely on any structural software, but perform the optimization independently and in most cases just in static conditions. In this sense, they are either site-specific or problem-specific, lacking generalization capability and transferability. Examples of optimization approaches in static conditions can be found in Sigmund [16], who devised and optimization methodology adopting MATLAB or in the work by Balling and colleagues [15]. The latter adopted a genetic algorithm (GA) to perform a more inclusive optimization (i.e., size, shape and topology) on skeletal structures but yet in a static load condition. Another application of GA can be found in the work by Rajan [17], focusing on the optimization of trusses.

This paper will present a simple, versatile and scalable methodology for the integration of Robot with MATLAB, with the primary purpose of automating structural behavior analysis and the potential for structural optimization. The authors would like to highlight that the proposed methodology is not meant to replace the engineers' judgement with an automated procedure, instead providing a tool that could be functional for reducing the amount of repeated calculations and allowing more time for the designer to investigate the technical solutions.

45.2 Methodology

45.2.1 Methodology Overview

The proposed methodology is presented, taking framed structure design as an example, where RC is considered as the structural typology. The reason behind this choice lies in the wide adoption of RC both in new and existing buildings, also consisting in a relatively more convenient technique both in terms of costs and employability. This approach can be adopted for both existing and new building and specifically in the second case, a preliminary dimensioning of the structure is needed as normally happens in the design practice prior to virtual modelling.

In detail, the methodology relies on the adoption of MATLAB in order to invoke systematically different Robot API components either to manipulate objects or perform specific operations, such as structural analysis (e.g., linear and non-linear).The overall structure of the API is briefly schematized in Fig. 45.1, including the main categories (e.g., Project,

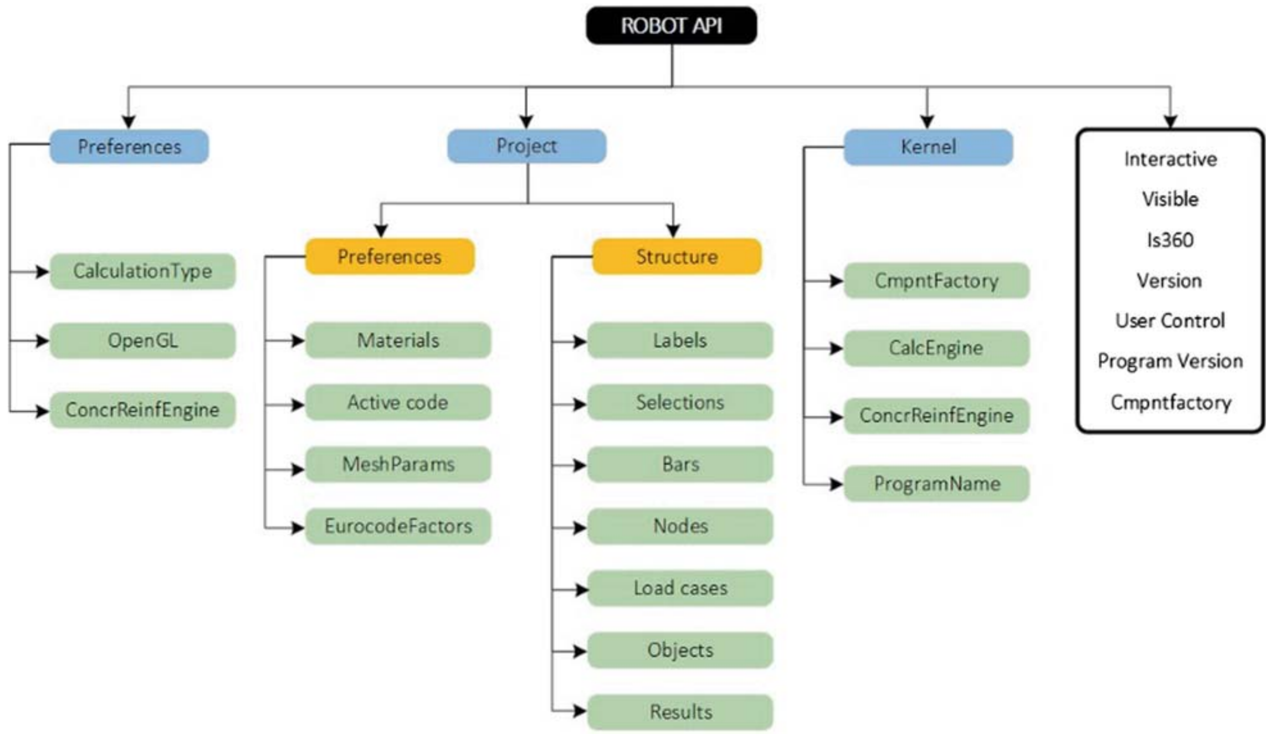


Fig. 45.1 Robot API main components

Preferences, Kernel) as well as the concerning subcategories (e.g., Preferences and Structure) adopting their original names. Figure 45.2 provides a comprehensive overview of the methodology. For each process within the methodology, the corresponding API component is also exemplified for performing specific tasks. The first step consists in the identification of the optimization objectives, which pertains both performance-related factors (e.g., displacement, deflection, buckling, distance between mass and stiffness centers) and costs. Examples of the design parameters that can be tweaked in the optimization phase to meet the targets can be identified in the following:

- Structural members' sections size;
- Characteristic resistance of concrete (R_{ck});
- Permanent non-structural loads on beams and slabs, since this feature reflects design choices such as partition walls and flooring;
- Steel class for reinforcement and type of reinforcement.

Following to the identification of the optimization targets, the user is asked to input a set of information needed to model the structure and perform the analysis. These dataset of information includes for instance the geometric data resulting from the preliminary dimensioning of the structure (e.g., interstorey, in-plan dimensions, storey number, section types), the hypothesized concrete and steel classes, loads to be considered (e.g., snow, wind, permanent, non-permanent and live loads). Grounding on these information the MATLAB code is able to remotely provide a graphic representation of the structure in Robot, combining the loads according to the chosen building regulation and consequently perform a linear dynamic analysis with response spectrum (i.e., modal analysis). Accessing the "results" section of the Robot API allows to tailor the output according to the typology of parameter to consider and then verify the desired elements.

Regarding the building modelling part, it is worth mentioning that some specific arrangements have been adopted in the modelling and calculation phases:

- Rigid diaphragms are introduced as horizontal structures, in accordance to what prescribed by Eurocode 8, §4.2.1.5 [3];
- Fixed constraints are imposed at the base of the building since just the superstructure's behavior is investigated;

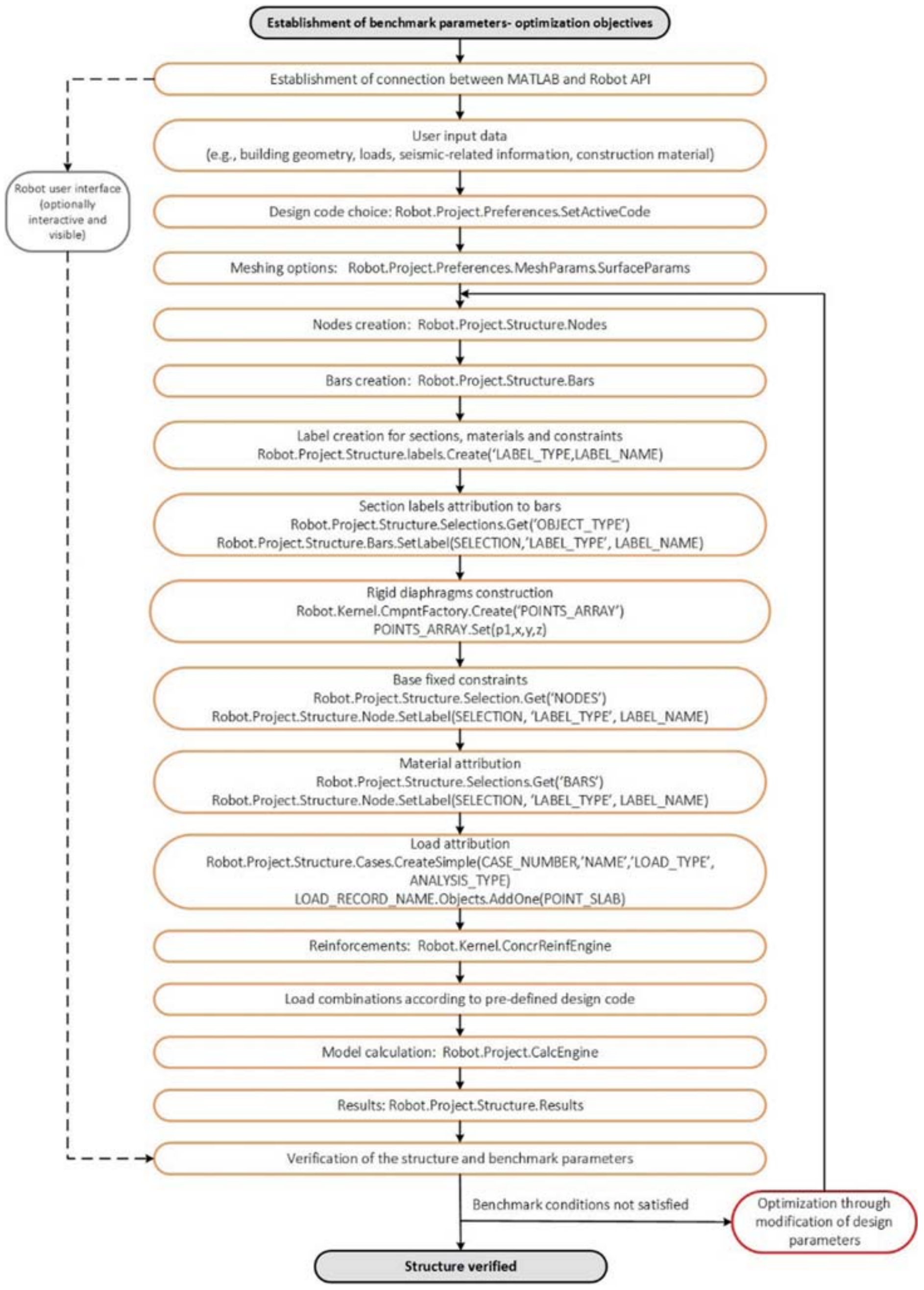
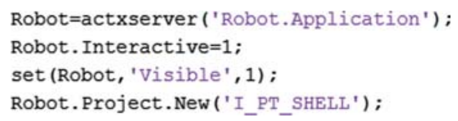

Establishment of benchmark parameters- optimization objectives
Establishment of connection between MATLAB and Robot API
User input data
(e.g., building geometry, loads, seismic-related information, construction material)
Robot user interface (optionally interactive and visible)
Design code choice: Robot.Project.Preferences.SetActiveCode
Meshing options: Robot.Project.Preferences.MeshParams.SurfaceParams
Nodes creation: Robot.Project.Structure.Nodes
Bars creation: Robot.Project.Structure.Bars
Label creation for sections, materials and constraints
Robot.Project.Structure.labels.Create('LABEL_TYPE,LABEL_NAME)
Section labels attribution to bars
Robot.Project.Structure.Selections.Get('OBJECT_TYPE')
Robot.Project.Structure.Bars.SetLabel(SELECTION,'LABEL_TYPE', LABEL_NAME)
Rigid diaphragms construction
Robot.Kernel.CmpntFactory.Create('POINTS_ARRAY')
POINTS_ARRAY.Set(p1,x,y,z)
Base fixed constraints
Robot.Project.Structure.Selection.Get('NODES')
Robot.Project.Structure.Node.SetLabel(SELECTION, 'LABEL_TYPE', LABEL_NAME)
Material attribution
Robot.Project.Structure.Selections.Get('BARS')
Robot.Project.Structure.Node.SetLabel(SELECTION, 'LABEL_TYPE', LABEL_NAME)
Load attribution
Robot.Project.Structure.Cases.CreateSimple(CASE_NUMBER,'NAME','LOAD_TYPE', ANALYSIS_TYPE)
LOAD_RECORD_NAME.Objects.AddOne(POINT_SLAB)
Reinforcements: Robot.Kernel.ConcrReinfEngine
Load combinations according to pre-defined design code
Model calculation: Robot.Project.CalcEngine
Results: Robot.Project.Structure.Results
Verification of the structure and benchmark parameters
Benchmark conditions not satisfied
Optimization through modification of design parameters
Structure verified

Fig. 45.2 Overall modelling methodology through Robot API in MATLAB

- Infill walls are modelled as linearly distributed loads over the beams;
- The flooring composition is computed through the application of permanent non-structural loads.

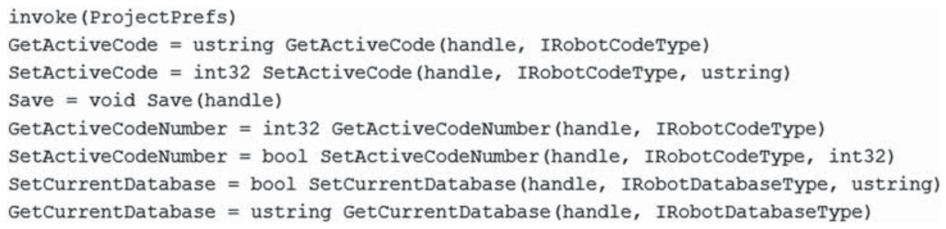
45.2.2 Integration Between Robot API and MATLAB

The dialogue between MATLAB and Robot relies on querying mechanisms that guide the path through the tree structure of the API itself invoking a specific function. Firstly, the Robot API is invoked from the MATLAB environment as in



```
Robot=actxserver('Robot.Application');
Robot.Interactive=1;
set(Robot,'Visible',1);
Robot.Project.New('I_PT_SHELL');
```

Fig. 45.3 Establishment of the connection between MATLAB and Robot API



```
invoke(ProjectPrefs)
GetActiveCode = ustring GetActiveCode(handle, IRobotCodeType)
SetActiveCode = int32 SetActiveCode(handle, IRobotCodeType, ustring)
Save = void Save(handle)
GetActiveCodeNumber = int32 GetActiveCodeNumber(handle, IRobotCodeType)
SetActiveCodeNumber = bool SetActiveCodeNumber(handle, IRobotCodeType, int32)
SetCurrentDatabase = bool SetCurrentDatabase(handle, IRobotDatabaseType, ustring)
GetCurrentDatabase = ustring GetCurrentDatabase(handle, IRobotDatabaseType)
```

Fig. 45.4 Example of use of invoke function for project preferences

Fig. 45.3, using ActiveX technology and renaming the querying process in the first line simply as "Robot" for ease of application in the rest of the script. This process will speed up the following steps in which the Robot.Application is invoked since the querying process follows the same path as in Fig. 45.1 and the different commands are separated by a dot. As an example, lines 2 and 4 in Fig. 45.3 clearly shows this process and its correspondence in Fig. 45.1, which guides MATLAB to access the desired command in the Robot API. Respectively, line 2 and 4 activate the User Interface of Robot and select the modelling typology. The options of making the Robot user interface visible and interactive are discretionary since the connection with the API already allows all the subsequent operations running in the background. To this regard it should be highlighted that when the Robot user interface is visible and interactive, any operation performed in one of the two environments (i.e., MATLAB and Robot) will reflect on the other one.

However, the set of operations available for each component of the API are documented in the Visual Basic, C++ and C# environments [13], but not for MATLAB so it has been of primary importance to apply the "invoke" function in order to get a precise description of the list of commands, including their input and output arguments. As an example, in Fig. 45.4 it is showed the result of the invoke function for the available operations on the Project Preferences. The corresponding path for this component, namely "Robot.Project.Preferences", has been named as "ProjectPrefs" for ease of adoption in the script. It is worth to point out that the function "invoke" will provide results when applied to "Objects" and not to variables resulting from other operations (e.g., double, string).

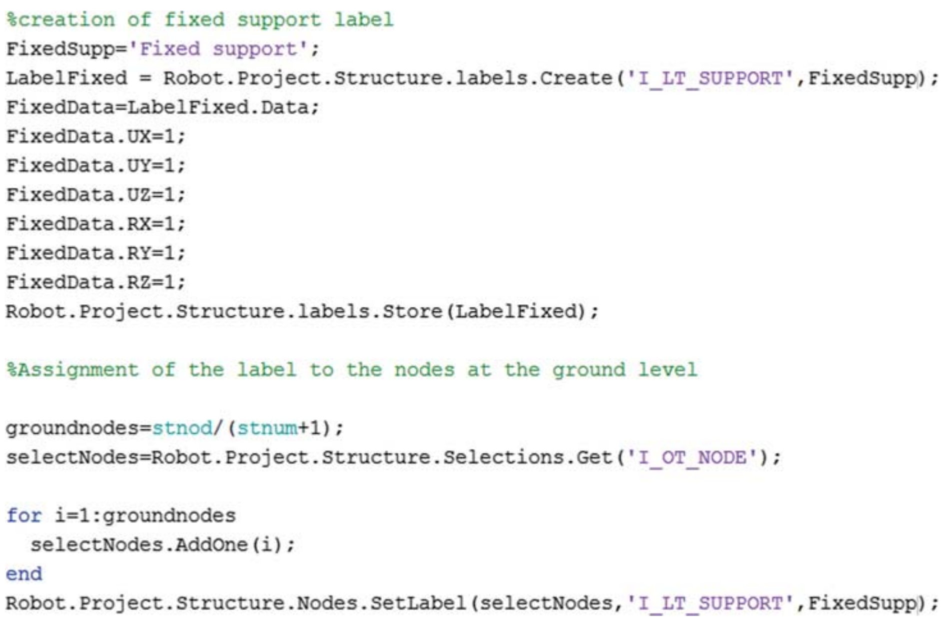
The first step consists in the definition of the structural grid based on the user input by creating the nodes as showed in Fig. 45.2. The nodes are functional for the successive step of bar creation that is analogous for beams and columns. Following to that, the creation of the labels allows to specify the properties that have to be assigned to the different structural members. The assignment procedure is carried out by adding the elements to characterize into a selection of a desired type of objects in the context of the specific label. To clarify it, Fig. 45.5 shows the example of the label creation of fixed support constraints (i.e., FixedSupp), the iterative addition of the nodes to the selection (i.e., selectNodes) and the assignment of the label to the latter in the last line.

The material attribution follows an analogous procedure, while the assignment of rigid diaphragms differs in the sense that, as showed in Fig. 45.2, there is the need of creating a point array corresponding to the panel perimeter through the Kernel. Following to that, labels for relevant properties such as thickness can easily assigned directly to the array.

After the overall geometry has been defined and characterized the following entails load definition and assignation to the desired elements. Each Load Case should contain its sequential number, a name for identification, the load type (e.g., permanent, live) and the type of analysis to be performed (i.e., dynamic modal in the example). Figure 45.6 shows in particular this process in the context of dead loads (i.e., self-weights) applied to the whole structure. After the definition of the load case it is necessary to specify the direction of application (z axis in this example) and the elements to which the loads has to be applied (i.e., entire structure).

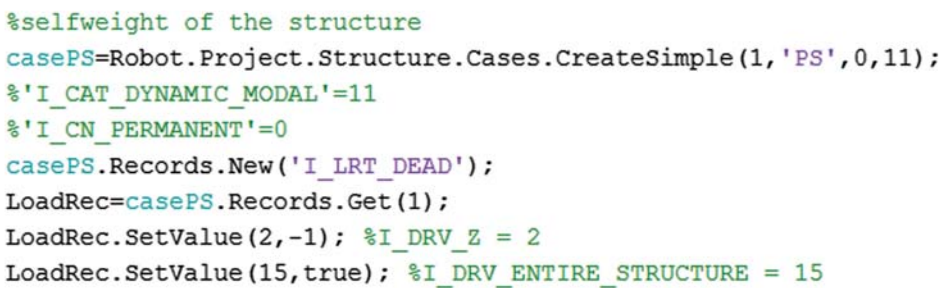
When the full characterization of the structure has been achieved, the calculation can be performed by referring to the CalcEngine component in the Kernel and successively results can be easily retrieved from the Results section as in Figs. 45.1 and 45.2. It is then possible to get, for instance, the value of the moment in a specific direction (e.g., x, y, z) for a specific beam at a certain point in its length and in relation to an established load scenario as shown in the example of Fig. 45.7.

Based on the results, verifications can be carried out in association with the compliance to the optimization targets. If the results of this step fulfill the objectives, then no further analysis is required, otherwise bespoke adjustments to the design



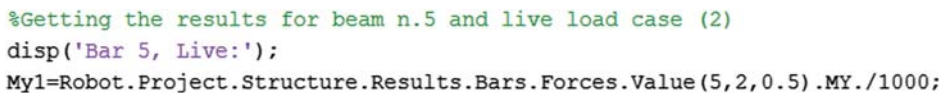
```
%creation of fixed support label
FixedSupp='Fixed support';
LabelFixed = Robot.Project.Structure.labels.Create('I_LT_SUPPORT',FixedSupp);
FixedData=LabelFixed.Data;
FixedData.UX=1;
FixedData.UY=1;
FixedData.UZ=1;
FixedData.RX=1;
FixedData.RY=1;
FixedData.RZ=1;
Robot.Project.Structure.labels.Store(LabelFixed);

%Assignment of the label to the nodes at the ground level

groundnodes=stnod/(stnum+1);
selectNodes=Robot.Project.Structure.Selections.Get('I_OT_NODE');

for i=1:groundnodes
  selectNodes.AddOne(i);
end
Robot.Project.Structure.Nodes.SetLabel(selectNodes,'I_LT_SUPPORT',FixedSupp);
```

Fig. 45.5 Label creation assignment for base fixed constraints in MATLAB



```
%selfweight of the structure
casePS=Robot.Project.Structure.Cases.CreateSimple(1,'PS',0,11);
%'I_CAT_DYNAMIC_MODAL'=11
%'I_CN_PERMANENT'=0
casePS.Records.New('I_LRT_DEAD');
LoadRec=casePS.Records.Get(1);
LoadRec.SetValue(2,-1); %I_DRV_Z = 2
LoadRec.SetValue(15,true); %I_DRV_ENTIRE_STRUCTURE = 15
```

Fig. 45.6 Label creation assignment for base fixed constraints in MATLAB



```
%Getting the results for beam n.5 and live load case (2)
disp('Bar 5, Live:');
My1=Robot.Project.Structure.Results.Bars.Forces.Value(5,2,0.5).MY./1000;
```

Fig. 45.7 Extrapolation of moment in y direction for a specific beam subjected to live load

parameters are made and the process is repeated iteratively till the complete satisfaction of the benchmark parameters. A suite of local (e.g., nonlinear programming) and global (e.g., genetic algorithm and particle swarm optimisation), constrained and unconstrained, and single-objective and multi-objective optimisation algorithms can be readily adopted to search for optimum design solutions with potentially conflicting objectives. This embodies the development of a MATLAB optimization client that progressively manipulates COM objects in the Robot simulation server through the use of ActiveX technologies. A local Robot simulation server is created to perform continuously structural design and analysis with selected design parameters iteratively tailored for a specific design scenario employing the MATLAB optimization toolboxes.

45.3 Case Study

The example structure presented in this paper is depicted in Fig. 45.8a and consists in a residential building in Old Beichuan County located in the Sichuan province, China. In detail, the building shows a six-storey RC frame structure with double-layer masonry infill walls. The clear extensive level of damage followed the 2008 Wenchuan seismic event, measuring 8.0 M_s. The process described above has been adopted to model the buildings as in Fig. 45.8b and in this section

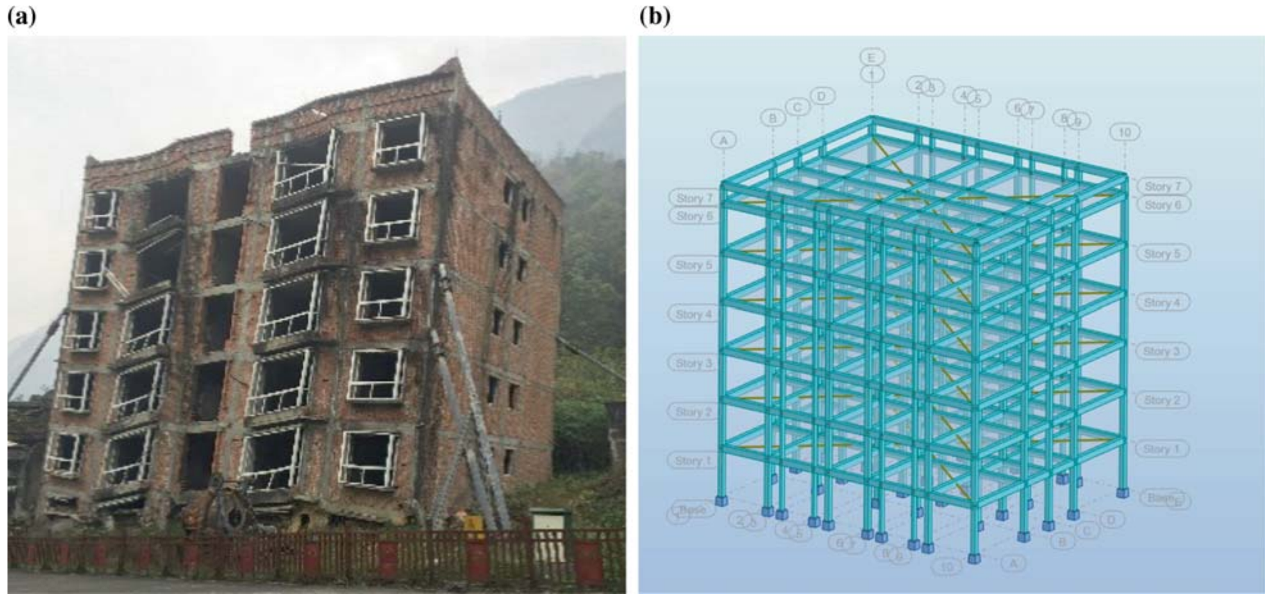
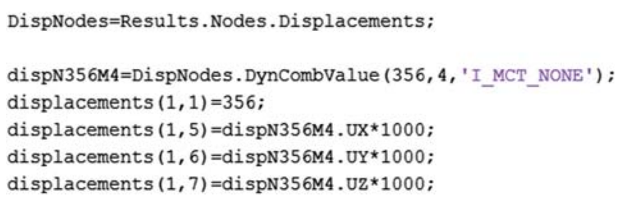


Fig. 45.8 State of the art (**a**) and Robot model (**b**) of a residential building in Old Beichuan County



```
DispNodes=Results.Nodes.Displacements;

dispN356M4=DispNodes.DynCombValue(356,4,'I_MCT_NONE');
displacements(1,1)=356;
displacements(1,5)=dispN356M4.UX*1000;
displacements(1,6)=dispN356M4.UY*1000;
displacements(1,7)=dispN356M4.UZ*1000;
```

Fig. 45.9 Residential building in Old Beichuan County

Table 45.1 Displacement data for accidental seismic combination at the different storey levels

Storey	Node number	UX (mm)	UY (mm)	UZ (mm)
Roof wall	356	1.520E+01	−1.302E−03	−2.409E+00
6	297	1.511E+01	3.204E−05	−2.402E+00
5	241	1.405E+01	2.414E−05	−2.311E+00
4	203	1.225E+01	1.880E−05	−2.090E+00
3	165	9.807E+00	1.350E−05	−1.745E+00
2	127	6.843E+00	8.166E−06	−1.280E+00
1	22	3.513E+00	2.773E−06	−6.978E−01
0	21	0	0	0

the focus is on how to retrieve the results and their interpretation in a real-case scenario. A preliminary assessment of the damage mechanism shows clear differential settlements, probably due to a liquefaction-prone soil combined with a superficial foundation system that led to the complete collapse of the ground floor storey. Given the lack of geotechnical information, only the superstructure is modelled and hence fixed constraints are placed at the base of the building, in order to isolate the uncertainties brought by soil-related aspects.

The displacements are collected through the Results server accessible through the path visible in Fig. 45.1, and hence consisting namely in moving from the Robot.Application main component, to the Project subsection and then to Structure, in which the section Results is located. Figure 45.9 shows in detail the use of the DynCombValue function, which allows to get absolute displacements (i.e., UX, UY and UZ) for a specific node of the structure (i.e., 356 in the example), given a certain combination (i.e., 4, accidental seismic). Another parameter to implement is the mode of combination adopted (e.g., CQC, SRSS) while more than one modes are presents, specifically in modal analysis. In this example it is presented the case of accidental seismic combination, which has been proved to be the more significant in terms of displacements. In Table 45.1 these data are listed in detail considering one node per each floor. This is allowed since the adoption of rigid diaphragms prevents relative displacements between points pertaining to the same diaphragm.

The displacement increases with the height although achieving a maximum value of about 1.2–1.5 cm in X direction, corresponding to the axis parallel to the longest side of the building. The node numbered as 21 shows no displacement because it is one of the fixed points at the base of the structure. In light of the preliminary load analysis and these results, the structure performed poorly due to the excessive weight of non-structural elements such as infill walls that intensified the differential settlements caused by the soil liquefaction. Hence, the seismic action alone did not trigger the collapse mechanism, but it contributed in combination to unsuitable design choices, such as the foundation system and the adoption of a heavy infill wall typology.

45.4 Conclusion

The paper focuses on a novel methodology for performing structural behavior analysis adopting MATLAB to access the Robot API and progressively manipulate objects. Different types of analysis can be performed and in this research specifically a RC framed structure has been analyzed through linear dynamic calculation. The overall process has been presented, from the connection between MATLAB and the Robot API to the extrapolation of displacement data, as an example. Future work will entail the implementation of multi objective optimization and different types of materials in order to embrace a wider percentage of design.

References

1. Cerè, G., Rezgui, Y., Zhao, W.: Critical review of existing built environment resilience frameworks: Directions for future research. Int. J. Disaster Risk Reduct. **25**, 173–189 (2017)
2. Fisher, A., Sharma, S.: Exploiting Autodesk Robot Structural Analysis Professional API for Structural Optimization, http://bimandbeam.typepad.com/. Last accessed 26 Apr. 2018
3. The European Union Per Regulation 305/2011, EN 1998-1:2004+A1:2003: Eurocode 8: Design of structures for earthquake resistance, Brussels (2004)
4. FEMA: 2015 NEHRP. Recommended Seismic Provisions: Design Examples, Washington D.C. (2016)
5. Adaptsoft, ADAPT Structural Concrete Software (2017)
6. Tekla, Tekla Structures. Administrator's Release Notes (2015)
7. Computers & Structures Inc, SAP 2000 User Manual, Version 19 (2016)
8. Autodesk, Autodesk Robot Structural Analysis Professional 2010 Training Manual (2009)
9. Autodesk, Autodesk Robot Structural Analysis. Metric Getting Started Guide (2010)
10. Collette, Y., Siarry, P.: Multiobjective optimization: principles and case studies, 1st edn. Springer, Berlin (2003)
11. MathWorks, MATLAB Optimization Toolbox, https://uk.mathworks.com/products/optimization.html. Last accessed 18 Apr. 2018
12. Dolsek, M.: Development of computing environment for the seismic performance assessment of reinforced concrete frames by using simplified nonlinear models. Bull. Earthq. Eng. **8**, 1309–1329 (2010)
13. Autodesk, Autodesk Robot Structural Analysis Professional 2018. Robot Object Model (2017)
14. Baldock, R.: Structural optimisation in building design practice: case-studies in topology optimisation of bracing systems. University of Cambridge (2007)
15. Balling, R.J., Briggs, R.R., Gillman, K.: Multiple optimum size/shape/topology designs for skeletal structures using a genetic algorithm. J. Struct. Eng. **132**(7), 1158–1165 (2006)
16. Sigmund, O.: A 99 line topology optimization code written in Matlab. Struct. Multi. Optim. **21**(2), 120–127 (2001)
17. Rajan, S.D.: Sizing, shape, and topology design optimization of trusses using genetic algorithm. J. Struct. Eng. **121**(10), 1480–1487 (1995)