計算数理工学論文集 Vol. 21 (2021 年 12 月), 論文 No. 10-211218 JASCOME

# WEB-BASED VISUALIZATION FRAMEWORK ON FREEFEM

Yu-Hsun LEE[1] and 藤原 宏志[2]

Yu-Hsun LEE and Hiroshi FUJIWARA

1) 京都大学大学院 情報学研究科 (〒 606-8501 京都市左京区吉田本町, E-mail: andylee@acs.i.kyoto-u.ac.jp)
2) 京都大学大学院 情報学研究科 (〒 606-8501 京都市左京区吉田本町, E-mail: fujiwara@acs.i.kyoto-u.ac.jp)

FreeFEM is a friendly integrated numerical computation software for partial differential equations in 2D and 3D based on the finite element methods. In the present paper, we introduce an intuitive web interface for the visualization of numerical results calculated by FreeFEM. With our web-based user interface, the users can easily access data interactivity and explore customizable graphics on modern browsers such as Edge, Safari, and Chrome. Also, the users can interact with a live render animation for problems with time evolution. We include a Hypertext Transfer Protocol (HTTP) server package in our FreeFEM dynamic loading module written in the programming language C++.

***Key Words***: FreeFEM, Visualization, Client-Server Model, Domain Decomposition Method, Parallel Computation, CAE, Education of FEM

## 1. Introduction

In the present paper, we propose a novel visualization framework on FreeFEM[(1)], which is one of the integrated development environments for the finite element methods (FEM). Since the default FreeFEM visualization assumes that the program runs on the local machine, various network and security configurations are required to visualize numerical results obtained by FreeFEM on remote computers. In order to overcome the drawbacks, we adopt web technology to improve the usability of interactive visualization in large-scale computation and educational purposes. The present study shows design and implementation of interactive visualization of numerical results via client-server architecture. It depends only on open source libraries and its framework is easily applicable to other numerical simulation environments.

FEM is a method for approximately solving differential equations by discretizing function spaces. By virtue of its mathematical foundations, it has been widely used not only in research and development but also in education. In recent years, the domain decomposition method (DDM) and parallel linear solvers have been commonly employed for large-scale computations appeared in advanced research and development. Several commercial [(2)(3)] and non-commercial [(1)(4)] environments are available, and they are commonly equipped with mesh generation on a domain of the problem, numerical integrations, linear solvers, and visualization of results. In other words, they enable us to concentrate on the input and output of the problems of interest by FEM.

FreeFEM is an open-source FEM package mainly developed by J. L. Lions Laboratory in France. Since it provides a simple original script language designed for FEM computations, users can specify domains, weak forms, and solvers which form the central roles and characteristics of FEM without precise knowledge of scientific programming languages and parallel computation by Message Passing Interface (MPI) or OpenMP. However, its output directly to the screen is assumed to be done on the local machine. Therefore, it is impossible to display numerical results on the local screen obtained with FreeFEM executed on remote machines such as supercomputers. Moreover, the visualization tool `ffglut` equipped with FreeFEM is based on OpenGL 1.0, which is inefficient and outdated. Main development team of FreeFEM are also planning to replace `ffglut` with new client `FFGraphics`[(5, 6)] which is developed with Vulkan[(7)].

The purpose of this study is an improvement of FreeFEM output issues. To this end, we adopt a client-server architecture. It also provides a Web-based User Interface (WUI) to control output information. Compare than the new client

2021 年 10 月 8 日受付, 2021 年 11 月 15 日受理

FFGraphics proposed by FreeFEM team, WUI has much more mobility than desktop client and the users do not need to install any extra package on their computers or mobile devices. By exploiting this feature, the web server function is integrated in the proposed module, and thus only standard browsers are required to visualize the results in the proposed framework. Our implementation is available from GitHub[8], and consists of two parts; one is a dynamic loading module in FreeFEM on the server-side, and the other is a web application for rendering data in the modern browser on client machines. Multiple users can simultaneously connect to the server provided by the module, and they can choose output information independently. For understanding FEM output, isolines and bird's eye view are presented. In order to facilitate a multifaceted understanding of FEM computations, users can see mesh numbering and DDM status. That is, our presented framework is also suitable for educational purposes.

In the next section, we show the architecture of the web-based framework. The third section is devoted to the usage of the dynamic module and web interface. Then we will introduce the visualization of MPI. Finally, we will show visualization in 3D computation.

## 2. Architecture

The proposed client-server architecture enables users to share the same data among different platforms. For cluster computing, the users can visualize the result immediately by the browsers without downloading data. For educational purposes, students and instructors can interact with the numerical result and obtain an intuitive understanding.

To implement the web-based interface, we use a client-server architecture in this work. We include a cross-platform non-blocking HTTP server-Mongoose[9] through the dynamic loading interface provided by FreeFEM. Mongoose is the most popular networking library on Github and used by various open source and commercial products[9] which ensure the usability of the client-server communication. Over the proposed HTTP server, we built several Web Application Programming Interfaces (APIs) to communicate with the data generated by FreeFEM. Moreover, we implement the framework under multithreading architecture. Thus users can check the visualization results in real-time during time-stepping computation, which is introduced in Fig. 3. For the communication between server and client, we use the gzip library `libz`[10] to compress the file size of numerical results at most 95% and improve the data transmission performance.

In the client part, modern web browsers are available to receive and process the transferred data. For the 2D images, we use the HTML5 canvas element for raster-based rendering, which provides better performance with many meshes then rendering vector graphics. For those who want to store high resolution graphics, users can also switch it to Scalable Vector Graphics (SVG) mode provided by `svg.js`[11]. Table 1 shows comparison of HTML5 canvas mode and SVG mode.

Table 1 Comparison between HTML5 canvas and SVG in 2D mode

| | HTML5 canvas | SVG |
|---|---|---|
| package | built-in | SVG.js |
| render method | raster-based | vector-based |
| performance (30x30 square mesh, 1800 triangles) | 258ms | 1075ms |

We also provide the bird's eye view for the scalar field data by using `three.js`[12], which is a cross-browser JavaScript library using Web Graphics Library (WebGL). The hidden surface removal is also processed by WebGL automatically. Both `svg.js` and `three.js` are well known graphics libraries in drawing 2D and 3D images.

## 3. Usage of the Web-based Framework

The proposed environments consists of 4 files and a folder:

- `webplot.cpp`: main source code
- `mongoose.c, mongoose.h`: Mongoose[9], a networking library for C/C++
- `macro_ddm.idp`: extended macro for MPI visualization
- `index.html`: main web application
- `static`: JavaScript packages and CSS files

The `webplot.cpp` is designed as a dynamic loading module of FreeFEM, and manipulates the data from FreeFEM and establishes web APIs communication over integrated Mongoose server. The `macro_ddm.idp` will be introduced in section 4. The proposed web application is implemented in `index.html` and included libraries like `svg.js` and `three.js` are in the `static` folder.

### 3.1. Compile

The proposed environments is available on GitHub[8]. To compile the module, the gzip library `libz`[10] should be linked. After downloading the source file, use the FreeFEM built-in command

```
$ ff-c++ mongoose.c webplot.cpp -lz
```

to compile the module. After the compilation, the dynamic module will be generated in the folder same as `webplot.cpp` (the filename of generated module on Linux platform will be `webplot.so`, on macOS will be `webplot.dylib`, and on Windows will be `webplot.dll`).

Moreover, the static file folder and `index.html` should be placed in the same directory with generated dynamic module `webplot.so`.

### 3.2. Usage of the Module

To use the web-based interface, the users have to load the proposed module to set up the server and APIs. Fig. 1 shows an example to pass the data through the `webplot()` function and the `server()` function to start the server.

```
load "webplot"
//load dynamic module

server();
//start server
border ba(t=0,1.0){x=t;   y=0;  label=1;};
border bb(t=0,0.5){x=1;   y=t;  label=1;};
border bc(t=0,0.5){x=1-t; y=0.5;label=1;};
border bd(t=0.5,1){x=0.5; y=t;  label=1;};
border be(t=0.5,1){x=1-t; y=1;  label=1;};
border bf(t=0.0,1){x=0;   y=1-t;label=1;};

mesh Th = buildmesh( ba(6)+bb(4)+bc(4)+
                     bd(4)+be(4)+bf(6) );

fespace Vh(Th,P1);
Vh u,v;
func f = 1;

solve Function(u,v)
  = int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))
    - int2d(Th)(f*v) + on(1,u=0);

webplot(Th);
webplot(u,Th);
//pass data through webplot()

show();
//keep server thread launching
```

Fig. 1 Usage of the web-based module in FreeFEM `edp` file

Users can set following options to setup the server with `server()` command, and the value inside brackets [ ] is the default value:

- host[`127.0.0.1`] : run server on the IP address.
- port[`1234`] : run server on the port.

Note that the default IP address, 127.0.0.1, is only accessible from the local computer. In order to make the server accessible from other machines via network, the IP address of the server machine must be specified by the option. For example, when the server is accessible through 192.168.2.1 with port 8000, the users can set the options with the following code. The architecture of remote computing with these example options is shown in Fig. 2.

```
server(host="192.168.2.1",port=8000)
//or
server(host="0.0.0.0",port=8000)
```

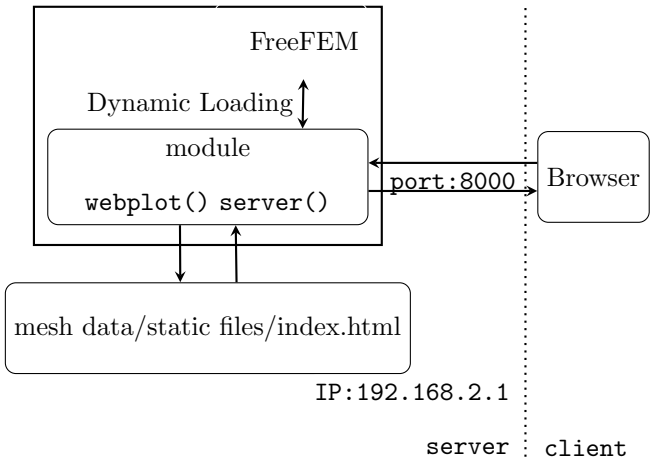


Fig. 2 Architecture of system under remote computing

For the options of figures, the users can setup the preferences of the figures on the web interfaces which will be introduced in the next subsection. The `webplot()` function is equipped with an option

- `cmm[""]` : comment shown on the graph.

For instance, in time evolution problems, if the users need to generate a series of plots with comments of specifying timestep, the `cmm` option provides the functionality to set the customize text as the example shown in Fig. 3.

### 3.3. Usage of the Web Interface

After the server has been started, the users can launch the web application on a browser via `http://<host>:<port>`. The left column of the interface provides some functionalities of specifying plotted materials and styles as shown in Fig. 4. The users can click the checkboxes to change the details such as index of mesh, isolines, labels of edges, etc. to

```
load "webplot"
server();
int TIMESTEPS=100;
...
for ( i = 1; i <= TIMESTEPS; i++) {
  ...
  webplot(phi,Th,cmm="time: t="+t);
}
show();
```

Fig. 3 Usage of `cmm` with a time evolution problem

show on the screen. The preview window is located on the right side and will render the figure with the user's preferences instantly. Fig. 5 shows the isolines and color bar, and Fig. 6 shows the mesh, vertices, and triangles indices and labels of edges. Fig. 7 and Fig. 8 show the bird's eye view with mesh or wireframe, whose view angle can be changed by mouse interactively. Users can also download the figure in `PNG` (available for 2D images and bird's views) or `SVG` (only for 2D images) formats as they see in the preview window. By default, we use the HTML5 canvas for raster-based rendering, and the users can also switch to SVG mode for vector graphics by clicking the "Preview in SVG" checkbox.

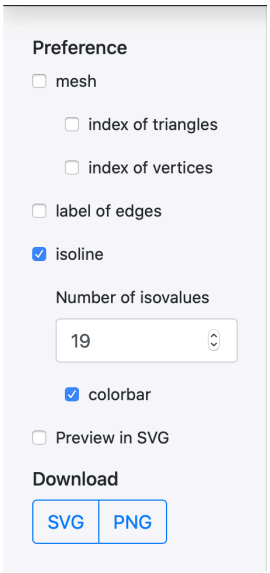


Fig. 4 Preference check boxes on web interface

## 4. Visualization of MPI

FreeFEM includes an additional macro `macro_ddm.idp` for domain decomposition methods which enables us to split the domain into several independent subdomains for parallel

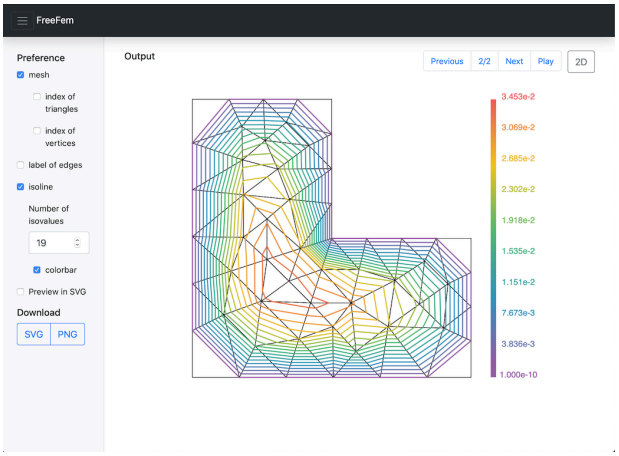


Fig. 5 Isolines as a result of Fig. 1. This is the default style.

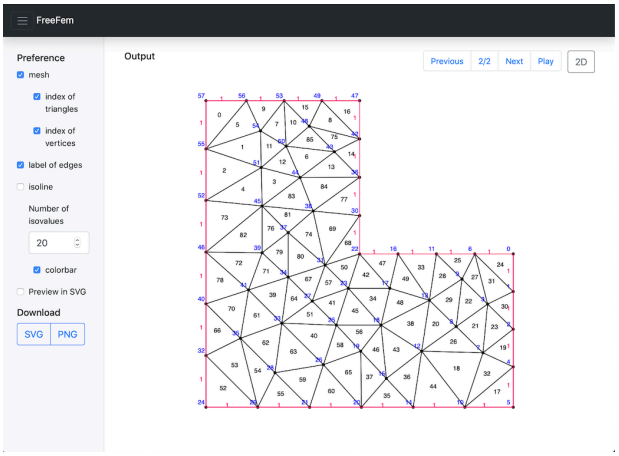


Fig. 6 Mesh indices and labels of edges (blue: vertex, black: triangle, red: edge label). Edge labels are specified in border instructions in Fig. 1

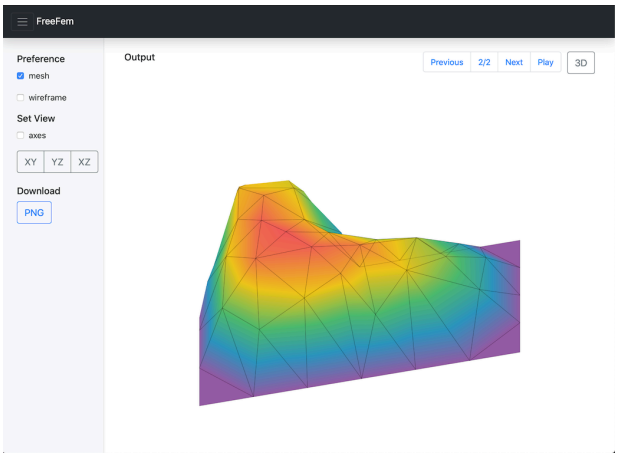


Fig. 7 Bird's eye views with mesh as a result of Fig. 1

computing. In order to visualize results of the computations, we extend the macro function `plotMPI()` in `macro_ddm.idp` to provide the MPI version of `webplot()` as shown in Fig. 9. For the `server()` function, we also implement the macro function `serverMPI()` to help the users to start the integrated server on the root process as shown in Fig. 10. The Fig. 11 and Fig. 12 depict the status of domain decomposi-

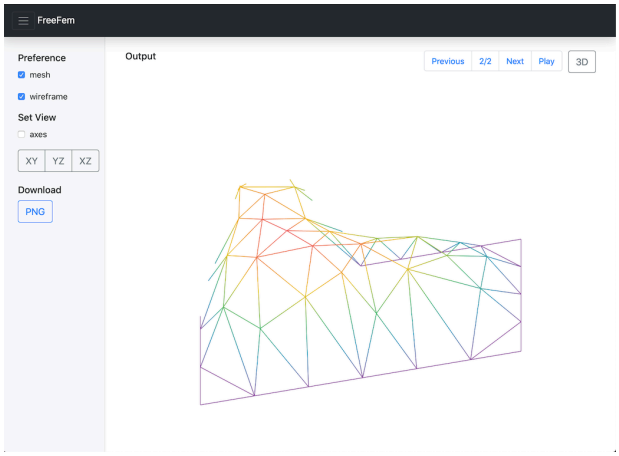


Fig. 8 Bird's eye views with colored wireframe as a result of Fig. 1

tion, and an example of the transient diffusion equation in FreeFEM library[13] respectively.

```
load "webplot"
macro plotMPI(Th, u, Pk, def, K, params)
...
if(mpirank == 0) {
  meshN[int] meshTab(mpisize);
  XhPlotPrivate<K>[int] def(uTab)(mpisize);
  ...
  for(int i = 0; i < mpisize; ++i) {
    webplotMPI(uTab[i],meshTab[i],
               i+1,mpisize,params);
  }
}
...
```

Fig. 9 Sample code of extended macro function `plotMPI()`

```
macro serverMPI(params)
if(mpirank == 0) {
    server(params);
}//
```

Fig. 10 Sample code of macro function `serverMPI()`

## 5. 3D Computations

FreeFEM can process with 3D volume mesh and boundry surface mesh. Similarly as stated in Section 2, our module supports these meshes. Fig. 13 shows an example of 3D volume mesh visualization. The users can set up the inter-

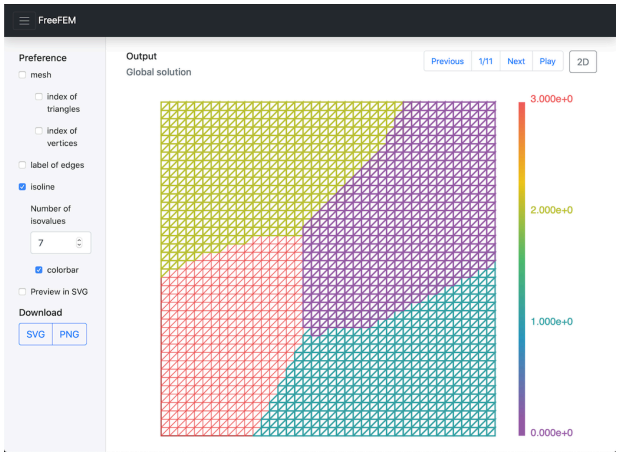


Fig. 11 Domain decomposition status with four processes generated by `plotMPI()` in Fig. 9

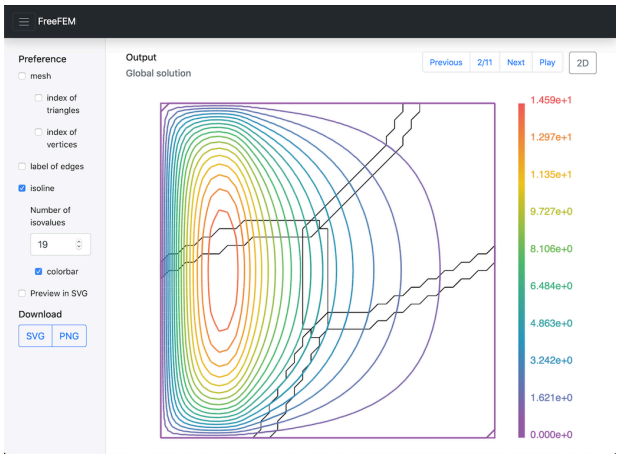


Fig. 12 Isolines of the numerical solution and domain decomposition status with four processes generated by `plotMPI()` in Fig. 9

section view of 3D results as Fig. 14 shows. Since 3D volume mesh contains a lot of tetrahedron elements, it is challenging to handle 3D model rendering for some mobile device. Thus, we still optimizing the visualization for 3D computations. For the MPI computation for 3D problems, similar to 2D MPI visualization, the macro function `plotMPI()` is also used.

## 6. Concluding remarks

In this paper, we discussed the web-based architecture for visualization on FreeFEM. The proposed framework is possible porting to other FEM computation software which provides Application Programming Interface to let us retrieve information such as numerical results and triangulation of the domain. The current distribution provides visualization of results of 2D and 3D computation and the status of domain decomposition for MPI computation. For demonstration, an online gallery is hosted at `https://freefem.andylee.tw`.

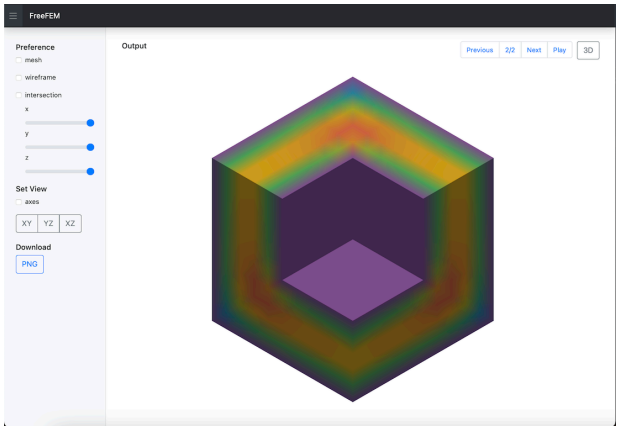


Fig. 13 Default style for 3D volume mesh

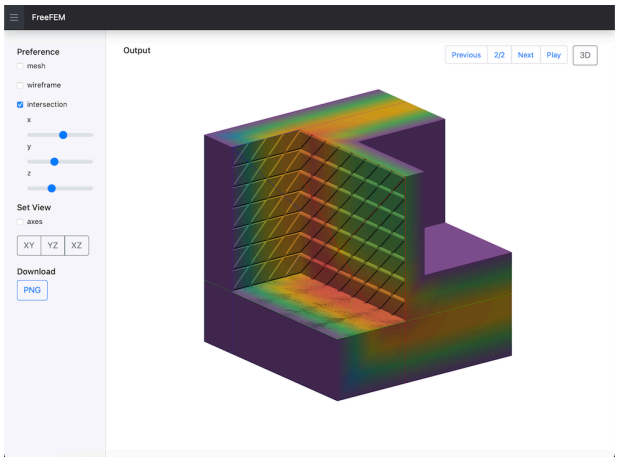


Fig. 14 Intersection view for 3D volume mesh

**Acknowledgment** This work was supported by JSPS KAKENHI Grant Numbers JP20H01821.

## 参考文献

(1) F. Hecht, New development in freefem++, J. Numer. Math., Vol. 20 (2012), 251–265.

(2) ABAQUS UNIFIED FEA, `https://www.3ds.com/products-services/simulia/products/abaqus/`

(3) Ansys: Engineering Simulation & 3D Design Software, `https://www.ansys.com`

(4) M. S. Alnaes, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes and G. N. Wells, The FEniCS Project Version 1.5, Archive of Numerical Software, vol. 3, 2015.

(5) F. Hecht, FreeFem-sources(feature-newplot), `https://github.com/FreeFem/FreeFem-sources/tree/feature-newplot`

(6) Q. Tessier, FreeFEM-graphic-client, `https://github.com/FreeFem/FreeFEM-graphic-client`

(7) Khronos Group, Vulkan, `https://www.vulkan.org`

(8) Y. Lee, freefem_webplot, `https://github.com/andylee830914/freefem_webplot`

(9) S. Lyubka, Mongoose, `https://www.cesanta.com`

(10) J. Gailly, M. Adler, zlib, `https://zlib.net`

(11) W. Fierens, svg.js, `https://svgjs.com`

(12) R. Cabello, three.js, `https://threejs.org`

(13) `heat-2d-PETSc.edp`, FreeFEM Documentation, `https://github.com/FreeFem/FreeFem-sources/tree/develop/examples/hpddm/heat-2d-PETSc.edp`