単一磁束量子回路のインタラクティブ可視化 ― 宇宙機での利用が期待されるデバイスに係る事例 ―

三浦 昭[*1]，松崎 恵一[*1]，石田 貴行[*2]，田中 雅光[*3]，井上 弘士[*4]

Interactive Visualization of Single Flux Quantum Circuits, a Device Expected to be on Board Spacecraft

MIURA Akira[*1], MATSUZAKI Keiichi[*1], ISHIDA Takayuki[*2], TANAKA Masamitsu[*3], INOUE Koji[*4]

ABSTRACT

This paper describes a method of interactive visualization for explaining the behavior of Single Flux Quantum (SFQ) circuits, a kind of logic circuit based on the principle of superconductivity.

For the space exploration in the future, there has been an idea to develop devices based on SFQ circuits onboard spacecraft. SFQ circuits are expected to have extremely low power consumption and high computational speed during operation. On the other hand, since the principle of SFQ circuits is greatly different from that of logic circuits of semiconductor devices such as CPUs and memories, visualization tools for understanding such type of circuits are expected to be useful.

In this paper we developed an interactive visualization application for SFQ circuits using JavaScript libraries including WebGL-based APIs and discussed issues as well as prospects.

Keywords: Single Flux Quantum (SFQ), Josephson transmission line, Visualization, Web Application

概　　要

超伝導の原理に基づく論理回路の一種である，単一磁束量子回路の動作を説明するためのインタラクティブ可視化手法について述べる．

宇宙探査に関わる将来構想のひとつとして，単一磁束量子回路を用いたデバイスを宇宙機に搭載することが検討されている．単一磁束量子回路はデバイスの動作に係る電力消費が極めて小さく，また演算速度の高速化も期待されている．一方で単一磁束量子回路は CPU やメモリ等，半導体デバイスの論理回路と動作原理が大きく異なるため，その理解の助けとなるような可視化ツールが有用であると期待される．

本稿においては，単一磁束量子回路の動作について，WebGL ベースの API 等の JavaScript を用いたインタラクティブ可視化アプリケーションを構築し，その課題や将来展望等について検討した．

[*] 2022 年 11 月 30 日受付 (Received November 30, 2022)
[*1] 宇宙科学研究所 (Institute of Space and Astronautical Science)
[*2] 研究開発部門 (Research and Development Directorate)
[*3] 名古屋大学大学院工学研究科 (Graduate School of Engineering, Nagoya University)
[*4] 九州大学大学院システム情報科学研究院 (Faculty of Information Science and Electrical Engineering, Kyushu University)

1. はじめに

筆者らは, 様々な宇宙科学データや宇宙探査に関わる可視化に取り組む中で, 具体化されていない, 研究レベルの将来の計画や構想等に関しても, このような可視化に対する需要が少なくないことを認識した.

そのような将来構想のひとつとして, 単一磁束量子回路を用いたデバイスを宇宙機に搭載することが検討されている. 単一磁束量子回路は, 超伝導の性質を利用した論理回路の一種で, CPU やメモリ等, 半導体デバイスの論理回路と比べて, デバイスの動作に係る電力消費が極めて小さく, また演算速度の高速化も期待されている.

一方で単一磁束量子回路は CPU やメモリ等, 半導体デバイスの論理回路とは動作原理が異なるため, 初心者が同回路の振る舞いを理解することは必ずしも容易ではない. 同回路の動作原理について, 可視化アプリケーション等, 初心者が容易に取り組めるような教材が必要であると考えられる. また係る可視化は, 初心者に限らず, 回路の動作原理に対する理解を深める上でも意義があると考えられる.

そこで筆者らは, これまで宇宙科学データの可視化に取り組む過程で得た知見等 [1]にも鑑みつつ, 単一磁束量子回路のインタラクティブ可視化を試みた.

以下, 本稿においては, 単一磁束量子回路に係る可視化の検討過程における, 可視化要素の選定や 3D 表現の手法等について述べた上で, 実装した GUI の実行例や課題, 今後の展望等について述べる.

2. 単一磁束量子回路

2.1. 単一磁束量子

超伝導リングの中を通る磁束は, 式1に示すΦ_0の整数倍に量子化されることが知られている. その概念図を図 1 左に示す.

$$\Phi_0 = 2.0678 \times 10^{-15}[\mathrm{Wb}] \qquad (1)$$

このリングに, 超伝導ではない薄い層（ジョセフソン接合）を差し挟むことを考える. その概念図を図 1 右に示す. このようなジョセフソン接合を含む超伝導リングの場合, リングを流れる電流の位相差は, ジョセフソン接合部分も含めて2πの整数倍となり, リング内を通過する磁束に相当するもの（fluxoid）が量子化されると考えることができる.

この量子化された最小単位は「単一磁束量子」（Single Flux Quantum; SFQ）と呼ばれる.

単一磁束量子回路（SFQ 回路）は, この単一磁束量子をビットの 0, 1 に見立てて各種のデータ処理を行うものである.

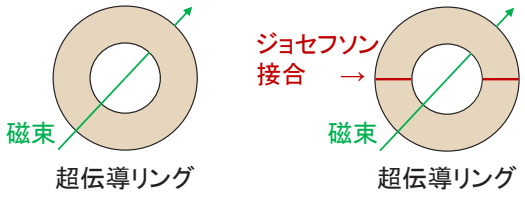

図 1 超伝導リングとジョセフソン接合

2.2. ジョセフソン伝送路

ジョセフソン伝送路は, SFQ 回路の一種である. 図 2 に, その概念図を示す. ジョセフソン接合を挟みつつ超伝導リングが直列に配置された構成となる. ビットの 1, 0 は, 各リングに SFQ が在るかどうかで区別される.

適切に回路を構成すると, ジョセフソン伝送路上を超高速で SFQ が伝搬するようになる.

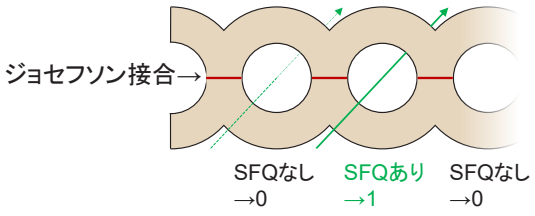

図 2 ジョセフソン伝送路

2.3. SFQ 回路を理解する上での課題

本節では, SFQ 回路を理解する上での課題を, CPU やメモリ等, 半導体デバイスの論理回路と比較しながら述べる.

2.3.1. 半導体デバイスの論理回路

半導体デバイスの論理回路は, トランジスタ（On・Off が切り替わる, 3 端子スイッチ素子）

等の組み合わせで構成される. 大規模集積回路は金属酸化膜半導体電界効果トランジスタ（Metal-Oxide-Semiconductor Field Effect Transistor; MOSFET）を用いた CMOS（Complementary Metal-Oxide-Semiconductor）が主流であり，例えば近年の CPU であれば，動作電圧は 1V 程度，動作周波数は数 GHz 程度のものまで見受けられる．その一方で小規模の IC や，論理回路の構成をプログラム可能な集積回路（Field Programmable Gate Array; FPGA）などを用いて，論理回路の組み立てや動作確認ができる環境を用意することも難しくない.

このような論理回路は，トランジスタの On・Off もしくは電圧の高い・低いがビットの 1・0 に紐づけられている. 外部から状態を把握する上でも, 測定された電圧値からビットの状態を容易に識別可能であり, 直感的にもわかりやすいものであると考えられる． 例えば定常的にビットを保持している状態であれば, 安価なテスター類で状態を確認可能であり, 動的に変化している場合も,（オシロスコープの性能・機能や動作クロック，プローブの挿入可能点等，諸々の制約はあるが）オシロスコープ等を用いてその変化を追跡することが可能である．これはまた，ビットの 1 を保持するためには電圧が発生し続けることも意味する.

2.3.2. SFQ 回路

SFQ 回路は，ジョセフソン接合という 2 端子素子が電流に基づいて動作する回路となる.

係る分野の用語を用いて説明するならば, SFQ 回路における本質的な物理量は, 超伝導波動関数の位相差である. 回路を構成するジョセフソン接合は, そこに流れる電流が閾値を超えてスイッチすると位相差が 2π 変化し，保持していたビット情報（SFQ の状態）が他のリングへと移動する. 上記半導体デバイスの論理回路で測定される電圧がそのままビット情報に対応していると考えられるのに対して，SFQ 回路で測定される電圧は，位相差の時間微分に対応する値となる（詳細は後節の回路方程式に記す）．電圧が発生するのは，SFQ が伝搬する（ビットの情報が変化する）瞬間のみであり， 1mV オーダの電圧パルスがピコ秒（サブ THz）オーダという極めて短時間の変化として現れる. 各リングが定常的にビット情報を保持している間は電圧が発生しない. この性質は，係る消費電力を考えるならば， 半導体デバイスの論理回路に比べて低電圧であり, かつ極めて短時間しか電圧が発生しないため, 低電力となるメリットがある.

しかしながら, その動作を理解するという立場で考えるならば,「波動関数」や「位相」等，直感的に理解することが難しい用語が多く, 波動関数の位相差を直接観測することもできない. また微量かつ瞬間的な電圧変化の測定値から, ビット情報の 1・0 を判定することも必ずしも容易ではない．このような現状に鑑み，筆者らは仮想的に SFQ 回路の動作を再現し, インタラクティブに可視化する手法の必要性を認識するに至った.

3. 回路のモデル

SFQ 回路を理解するために, いくつかのモデルが提案されている．以下に，これまで試みられてきた, ジョセフソン接合を喩えるためのモデルと, その中から可視化対象として本稿にて選定した要素について述べる.

3.1. 振り子モデル

回路の振る舞いを振り子に喩えたモデルである．振り子モデルとしては，位相差が振り子（もしくは振り子の球が固定された滑車）の回転角に喩えられ, 振り子同士の連結は弾性のあるゴム紐等で喩えられる. 図 3 に, その例 [2]を引用する.

振り子モデルは位相差成分を振り子の動きとして直感的に表現できる利点があるため, 本稿においては，位相差を表現するために，振り子の回転角を可視化対象として選定した.

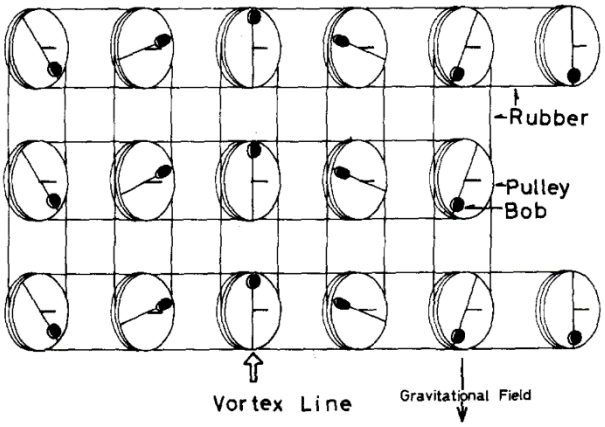

図 3 振り子モデルを説明した例 [2]

3.2. 等価回路

本稿で用いたジョセフソン伝送路を等価回路として表現したものを図 4，図 5 に示す．図 4 は簡略表現である [3]. 図 5 はジョセフソン接合箇所を詳細な要素に分割してモデル化したものである [4]．簡略表現でジョセフソン接合にまとめられていた箇所は，超伝導成分の他に，抵抗成分やキャパシタ成分が並列に接続された構成となっている．なお各図中の緑の×印は，図 2 に示したような SFQ のイメージを記したものである．

本稿においては，等価回路の電流・電圧の変化を可視化対象として選定した．

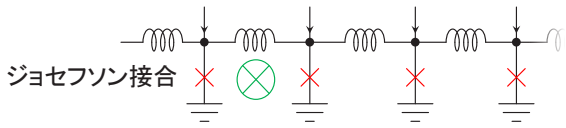

図 4 等価回路の例（簡略）

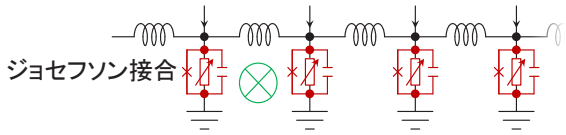

図 5 等価回路の例（詳細）

3.3. 洗濯板モデル

洗濯板モデルとしては, 粘性液体中の傾いた洗濯板の上を運動する質点として喩えられる. 図 6 に，その例 [5]を引用する．

同モデルを可視化する場合, 位相差を示す質点の移動を適切に追跡する必要があるが, これは他のモデルと比べて難易度が高いと判断し, 本稿において洗濯板モデルは可視化の対象外とした．

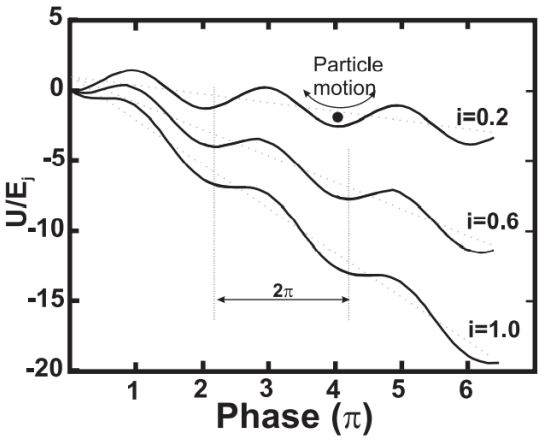

図 6 洗濯板モデルを説明した例 [5]

4. 可視化の概要

本稿においては, Web ブラウザ環境でインタラクティブな 3D 可視化を実現した．以下，可視化やデータ処理に係る主な要素と, 使用したライブラリ等について述べる．

4.1. 可視化要素の三次元配置

3 節で選定したモデルは，可視化要素に分割した上で, xeogl [6]を用いて 3 次元空間に配置する．ここで用いた xeogl は WebGL ベースの 3D CG ライブラリである

WebGL [7]は，Web ブラウザ上の JavaScript で利用できる，OpenGL 互換の 3D グラフィックス用 API である．WebGL を用いると，コンパイルやパッケージングの必要がなく, ソースコードや関連するデータ等を Web サイトにアップロードするだけで，様々なプラットフォーム上で 3D アプリケーションを共有できる．

可視化要素の配置概要を図 7 から図 10 に示す．これらの配置概要図は，入力電圧部分と，伝搬回路の 1 段分とを示している. 実際の可視化にあたっては, 指定された段数のオブジェクトをそれぞれ生成して配置する．

図 7 は，前節の図 4 に相当するオブジェクト配置，図 8 は図 5 に相当するオブジェクト配置を示している. 図中の電圧オブジェクトは正の値を赤，負の値を青で表示するものであり，同時に

2 つの色のオブジェクトが表示されることはない．電流は，順方向を橙，逆方向を水色で表示する．電流オブジェクトで囲まれた中には，磁束量子を模擬する半透明の円環を配置する．この円環もまた，順方向を橙，逆方向を水色で表示する．

図 9 は，振り子モデルに相当するオブジェクト配置を示している．これは位相差を表す振り子と，振り子同士の連結で構成される．文献 [2]においては振り子の球が円盤（滑車）に埋め込まれたようなモデルとなっているが，本稿における振り子は，見通しを良くするために，回転中心に位置する円盤と振り子の球を棒で結んだ構造としている．振り子同士の連結は，ゴムのような柔らかい表現ではなく，ばねを模した短冊状の板の連結で表現した．それぞれの短冊は隣接する短冊と密接しながらも不連続であり，また表裏の色を変えることで，そのねじれの程度を容易に視認できるようにした．

実際の可視化にあたっては，用意されたデータの種別に応じて，図 7，図 8 のいずれかのオブジェクトを図 9 のオブジェクトと組み合わせて表示する．図 10 は，図 8 と図 9 を組み合わせた配置例である．

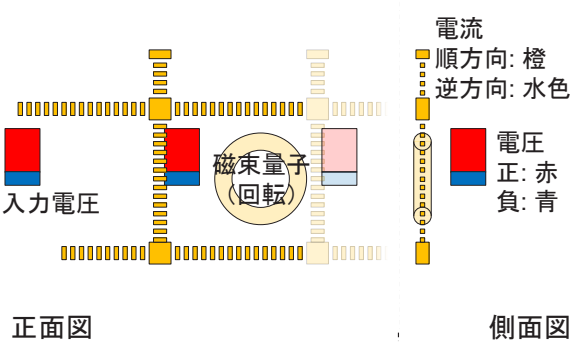

図 7 可視化要素の配置（等価回路，簡略）

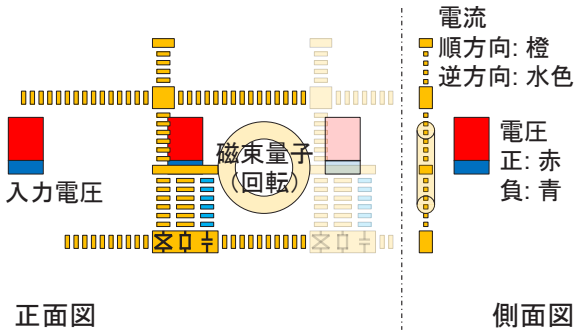

図 8 可視化要素の配置（等価回路，詳細）

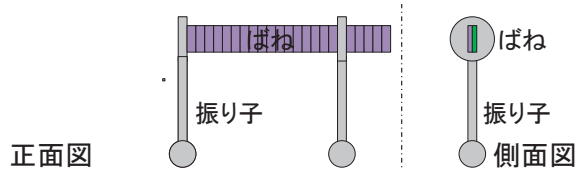

図 9 可視化要素の配置（振り子モデル）

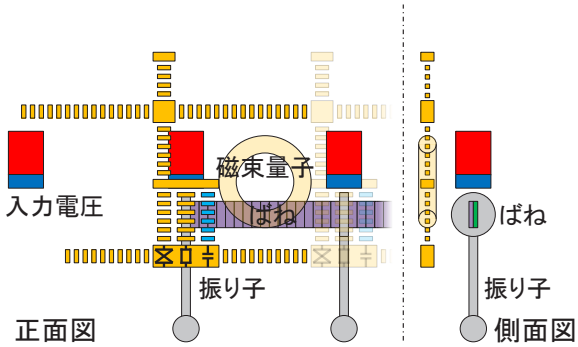

図 10 可視化要素の配置（総合）

4.2. メニュー生成

アプリケーション画面における，数値の表示や入力，項目選択，チェックボックス等のユーザインタフェース（以後，纏めて「メニュー」と記す）作成には，dat.gui [8]を利用した．

以下に，カメラ関連のパラメータ設定を例に，メニュー関連の機能について述べる．

図 11 にメニュー生成の例を示す．これは視点位置（X，Y，Z 座標）の変更可能範囲を-200 から 200 に設定した例である．同様にしてカメラ関連の諸設定や，その他の各種パラメータの設定も行う．ここで‘eyeX’，‘eyeY’，‘eyeZ’などは視点位置を示すキーワードとして例示したものであり，以後に例示するものと共通である．

```
var camera_menu = new CameraMenu;
gui.add(camera_menu, 'Camera_Preset', Object.keys(CameraPresets));
var cameraFolder = gui.addFolder('scene.Camera');

cameraFolder.add(camera_menu, 'eyeX',-200,200).listen();
cameraFolder.add(camera_menu, 'eyeY',-200,200).listen();
cameraFolder.add(camera_menu, 'eyeZ',-200,200).listen();
....
```

図 11 メニュー生成例（抜粋）

図 12 は，具体的なメニュー項目の操作を定義した例である. 各キーワードに対応したパラメータ類の現在値を, 対応するメニュー項目に設定する他, 前回更新時以降に利用者がプリセットの選択を変更した場合は, 与えられたプリセット値を用いてメニュー項目や対応するパラメータ類を更新する.

```
var CameraMenu = function () {
    this["Camera_Preset"] = "default";
    this.eyeX = camera.eye[0];
    this.eyeY = camera.eye[1];
    this.eyeZ = camera.eye[2];
    …
    var self = this;
    var lastPreset;
    var update = function () {
        var Preset = self["Camera_Preset"];
        if (lastPreset !== Preset) {
            lastPreset = Preset;
            var arg = CameraPresets[Preset];
            self["eyeX"] = arg.eyeX;
            self["eyeY"] = arg.eyeY;
            self["eyeZ"] = arg.eyeZ;
            camera.eye = [arg.eyeX, arg.eyeY, arg.eyeZ];
            …
        }
        self["eyeX"] = camera.eye[0];
        self["eyeY"] = camera.eye[1];
        self["eyeZ"] = camera.eye[2];
        …
        requestAnimationFrame(update);
    }
    update();
};
```

図 12 メニュー項目のアップデート例（抜粋）

図 13 は，プリセット値の設定例である．カメラパラメータの場合，視点・注視点等の値を幾つか事前に登録することで，拡大・縮小や視点変更等，視覚的な切り替えが容易になる.

```
var CameraPresets = [];
CameraPresets["default"] = {
    eyeX: -40,
    eyeY: 10,
    eyeZ: 30,
    lookX: 0,
    lookY: 0,
    lookZ: 0,
    upX: 0,
    upY: 1,
    upZ: 0
};
```

図 13 プリセット値の定義例（抜粋）

このような設定を, ここで例示したキーワード以外にも，必要に応じて書き加え，プログラム中の呼応するデータと連携させる.

また同時に表示される項目が増えすぎないように，メニューの展開や折り畳み等，dat.gui で提供される機能を用いて画面利用の効率化も図っている.

4.3. データ処理

4.3.1. 概要

位相や電圧, 電流等, 可視化に用いるデータは, 一旦配列に蓄えられた上で, 配列上の各時点の状態が可視化される. ソースコード中で係るデータ入出力の記述を簡素化するために，jQuery [9]の機能を利用した. これは xeogl や dat.gui とは異なり，GUI に陽に現れることはない.

4.3.2. データ入出力形式

可視化に用いるデータの入出力形式は，CSV 形式と JSON（JavaScript Object Notation） [10]形式を採用した.

CSV 形式は外部のデータ源を参照する際に有用である．本稿執筆時点では， 4.1 で配置された各可視化要素（電圧，電流，位相）に対応する値を外部の回路シミュレータにて事前計算した時系列データを読み込み, 前述の配列に格納するために CSV 形式を使用している．CSV 形式は簡素なフォーマットであり, 利用者の表計算ソフト等で容易に内容を確認することもできる. 一方で伝送路の段数や, 各カラムが示す値の種類等に変更があると，過去に生成された CSV データとの整合性が失われる恐れがある.

JSON 形式を用いると, JavaScript のオブジェクトを可読な形式で保持可能である．要素の追加・削除にも柔軟に対応できる反面, 表計算ソフトウェアや，JSON に対応していないアプリケーション等との互換性には難がある.

4.4. リアルタイム計算

4.4.1. 概要

4.3.2 に述べた手法で外部のデータ源を参照する場合，可視化される時系列データは，予め計算されて前述のデータ入出力形式で参照可能とな

っているものに限られる．この場合，利用者がパラメータを調整しながら回路の動作状況を確認することは困難である．（その都度外部の回路シミュレータを用いた計算結果を取り寄せる必要があり，現実的ではない）そこで本稿においては，次節に述べる回路方程式の数値解をリアルタイムに計算する，簡易の数値積分機能を実装し，利用者がパラメータ変更した際，それら変更結果が随時可視化に反映されるようにした．

数値積分のアルゴリズムは，オイラー法とルンゲ=クッタ法（RC4）を採用した．なおここに述べる数値積分機能は，本稿で対象としたジョセフソン伝送路に特化したものであり，汎用の数値積分機能ではない．

係る演算結果は，前節で述べた配列に一旦格納された上で可視化に供される．また前節で述べたデータ入出力機能により，配列に格納された各値を，CSV 形式もしくは JSON 形式で保存することも可能である．

4.4.2. 回路方程式

本稿で数値積分に用いた等価回路の方程式を式2～式5に示す．等価回路と各変数の対応を図 14 に示す．各段における位相差はθ_nと表す．ジョセフソン接合部分の各変数との対応は図 15 に示す．超伝導成分（図中の×印箇所）を流れる電流は，臨界電流I_cと位相差θ_nを用いて，$I_c \sin\theta_n$と表現される．抵抗成分Rについては，固定抵抗として扱っている．これらジョセフソン接合箇所の各変数やバイアス電流I_Bは，現時点では全段共通としている．

$$I_B + i_n - i_{n+1} = I_c \sin\theta_n + \frac{v_n}{R} + C\frac{dv_n}{dt} \tag{2}$$

$$v_n = \frac{\Phi_0}{2\pi}\frac{d\theta_n}{dt} \tag{3}$$

$$v_{n-1} = v_n + L_n\frac{di_n}{dt} \tag{4}$$

$$i_{N+1} = 0 \tag{5}$$

これらを式6～式9によって無次元化した式10～式13を数値積分の対象とする．

$$T = \frac{2\pi I_c R}{\Phi_0}t \tag{6}$$

$$I_n = \frac{i_n}{I_c} \tag{7}$$

$$V_n = \frac{v_n}{I_c R} \tag{8}$$

$$V_0 = \frac{v_{in}}{I_c R} \tag{9}$$

$$\frac{I_B}{I_c} + I_n - I_{n+1} = \sin\theta_n + V_n + \beta_c\frac{dV_n}{dT} \tag{10}$$

$$V_n = \frac{d\theta_n}{dT} \tag{11}$$

$$V_{n-1} = V_n + \frac{2\pi L_n I_c}{\Phi_0}\frac{dI_n}{dT} \tag{12}$$

$$\beta_c = \frac{2\pi I_c R^2 C}{\Phi_0} \tag{13}$$

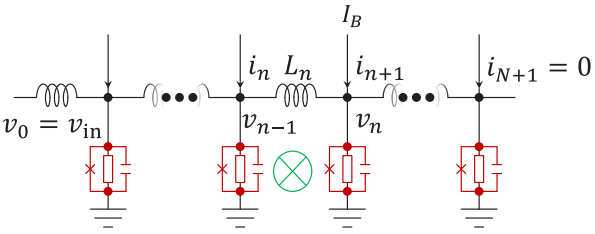

図 14 等価回路の数値積分

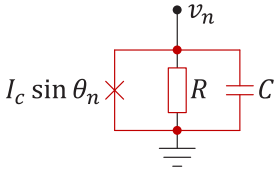

図 15 等価回路のジョセフソン接合部分

4.5. 実行例

以下に，構築した GUI の実行例を示す．

まず取り込みデータの可視化例を示す．これらは既存の回路シミュレータで実行したシミュレーション結果を取り込み，簡略モデルを用いて可視化した例である．

図 16 は，比較的オブジェクトの近くに視点を置いた例である．振り子の回転（位相差の変化）

に主眼を置いて，斜めから見るカメラワークとなっている．画面の右には各種パラメータのメニューが表示されている．上から順に，画面表示や I/O に関わる項目，カメラパラメータ等に関わる項目，外部データの読み込みに関わる項目等で構成される．これらのメニューは必要に応じてグループ化されており，グループ毎に展開・折り畳みが可能である．

図 17 はジョセフソン伝送路の全景を視野に収めた例である．SFQ の伝搬を概観するのに適した構図となっている．図 16 では折り畳まれていた，回路構成や表示速度等に関わる各種設定のメニューが展開されている．なお読み込まれたデータについて回路構成のメニュー項目を適切に表示するためには，シミュレーション結果のみでなく，シミュレーション時の各種パラメータも取り込む必要がある．

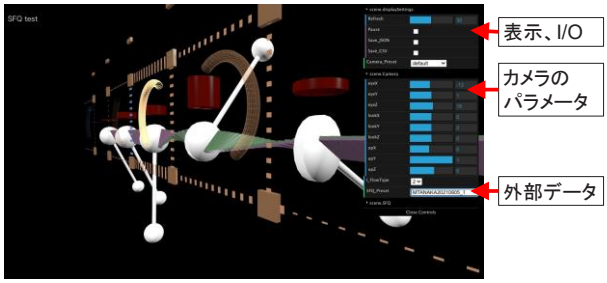

図 16 実行例（近景，取込データの可視化）

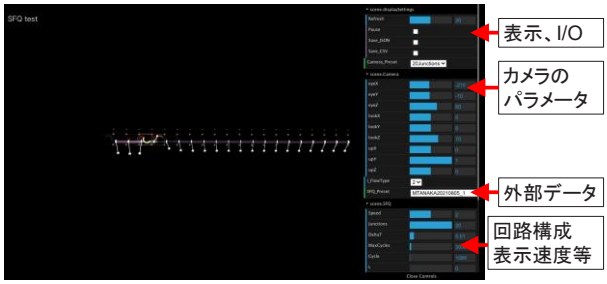

図 17 実行例（遠景，取込データの可視化）

続いて簡易の数値積分機能を用いた例を示す．図 18 は，詳細モデルを用いて数値積分結果を表示した例である．演算設定に関するメニュー項目は，現在の各パラメータの値を表示するとともに，利用者の入力に基づいたパラメータ変更も可能であり，変更されたパラメータは，即座に数値積分に反映される．

筆者らの体感としては，オイラー法，ルンゲ=クッタ法共に，数値積分に伴う待ち時間は殆ど感じられなかった．ただしこれは厳密に測定したものではなく，また実行環境によってもレスポンスが変化する可能性がある．

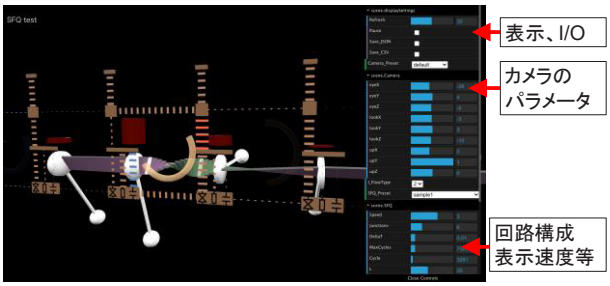

図 18 実行例（詳細モデル，数値積分結果）

5. 課題

このたびの可視化は，Web ブラウザ上で実行することを前提として開発を進めた．その結果として，利用者は特別なアプリケーションをダウンロードすることなく，様々な環境で可視化体験が可能となった．開発者側としても，一組のソースコード群を以て様々なクライアントに対応できる利点が得られた．

しかしながら，個々の OS やブラウザの組み合わせで動作確認した際には，完全に同一の利用体験が得られるとは限らないことも明らかになった．

5.1. レンダリングの速度

レンダリング速度はクライアント環境の種々性能によって左右される．これは CPU や GPU 等のハードウェア性能のみならず，同時実行されている他のアプリケーション類のリソース配分等の影響も受ける．そのため，執筆時点では，正確な時間変化率を再現するには至っていない．

5.2. 描画の問題

本稿における可視化では，一部の表現は線画で代替可能なものが含まれている．しかしながら多くのブラウザにおいて，ソースコードで指定した

線の太さがレンダリング時に正確に反映されない現象が見受けられたため, 現時点で線画による描写は, 半透明オブジェクトのエッジ描画等, 限られた用途にのみ採用している.

また仮想マシン上では, 動作が遅くなる以外に, 必ずしも意図した通りにレンダリングされない事例も見受けられた.

このような事例に鑑みると, かようなアプリケーション開発では, あらゆるクライアント環境での動作を保証することは容易ではない. 当面は, いくつかのクライアント環境を想定しつつ, それら環境において動作確認する等の対応が現実的であると考えられる.

5.3. 「わかりやすい」表現とは?

本稿における手法は, 振り子モデルと等価回路を組み合わせて, 様々な電流, 電圧や位相を可視化したものであるが, 振り子モデルと等価回路との関連性を明確に示す要素は追加されていない. これらの関連をわかりやすく表現するためには, さらなる工夫が必要であると考えられる.

また「わかりやすい」とは人それぞれであり, 開発者がわかりやすいと考える表現が, 必ずしも利用者にとってもわかりやすいとは限らない. 利用者からのフィードバックを受けつつ, わかりやすいアプリケーションを開発することが肝要とも考えられる.

5.3.1. 電流の表現

ひとつの例として, 電流の表現について記す. 本稿に述べた手法では, 各電流値を表現するにあたって, 当該オブジェクトの幅を変化させる表現と, オブジェクトの移動速度を変化させる表現の 2 種類を実装した.

オブジェクトの移動速度を変化させる場合, 抵抗成分を流れる電流オブジェクトの見かけの動きは, 位相差の変化, すなわち振り子モデルの回転変化と合致する. これを考慮しつつ, 本稿においては, 電流オブジェクトの移動速度や振り子の回転中心にある円盤のサイズ等を調整することで, 位相差の変化と電流オブジェクトの移動とが同期するような構成を取っている. ただしこの連動部分は必ずしも明確に視認できるレベルには至っておらず, さらなる改善が必要であると考えられる.

また今回の可視化モデルは電流を表すオブジェクトを直方体の等間隔な連なりで表現しているため, 電流値が大きい場合は電流オブジェクトの移動が早くなり, エイリアシングが発生する. これは見かけ上, 電流値に即したオブジェクト移動を表現できないことになる. 部分的に直方体のサイズを変える等, エイリアシング発生時にも電流の状況が把握できるような対策をとってはいるが, さらなる改善が必要と考えられる.

一方で電流値とオブジェクトの幅を比例させるような表現をした場合, オブジェクトのサイズで電流の程度を把握できるようになるが, その反面, 電流値（絶対値）が小さい時に電流のオブジェクトが視認し難くなる. 例えば入力電圧をかける前の初期状態などで, 等価回路のオブジェクトが一部欠けているように見えることがある.

5.3.2. 可視化要素のグラフ表示

このたび可視化対象とした各要素の時系列変化を正確に表現するのであれば, 2 次元グラフが有用であると考えられる. しかしながら 2 次元グラフを 3 次元空間に配置しつつ理解度を高めることは必ずしも容易ではない. オブジェクトとグラフを, 互いに干渉しないように別ウィンドウに表示することも選択肢の一つであるが, その場合, 3 次元空間での時系列変化と別ウィンドウでの時系列変化を, 利用者が同時に追跡することは容易ではなく, 回路の振る舞いをリアルタイムに可視化する場合, 必ずしもわかりやすい表現にはならないと考えられる.

6. おわりに

SFQ 回路のインタラクティブ可視化について, その概要と実行例, 課題について述べた. 可視化手法については, これで完成するものではなく,

実装方法や表現方法等, さらに工夫する余地があると考えられる.

現時点では, 段数の差こそあれ, 1 種類の回路についての可視化を実現したところであり, そのような回路に特化した実装となっている. 内蔵の数値積分やデータ入出力形式等も同様に, 1 種類の回路構成のみ考慮した実装となっている. 今後異なる構成の回路を可視化することを考慮するとなると, 回路の種別ごとにソースコードや入出力形式を考案するのではなく, 効率的に開発するための工夫が必要になると考えられる.

また冒頭に述べたように, 本稿の手法は, 宇宙科学データや宇宙探査に係る可視化を検討する過程で蓄積された知見等に由来するものがあり, ここで新たに蓄積された知見は, 翻って宇宙科学や宇宙探査等に関連する諸々の可視化手法への適用も期待されるところである.

参考文献

[1] 三浦 昭, “コンテンツ制作に適した宇宙科学データの活用方法,” *第 20 回情報科学技術フォーラム, I-028,* 2021.

[2] K. Nakajima, Y. Onodera, T. Nakamura , S. Risaburo, “Numerical analysis of vortex motion on Josephson structures,” *J. Appl. Phys., vol. 45, no. 9,* p. 4095-4099, 1974.

[3] K. K. Likharev , V. K. Semenov, “RSFQ Logic/Memory Family: A New Josephson-Junction Technology for Sub-Terahertz-Clock-Frequency Digital Systems, ” *IEEE Trans. Appl. Supercond. Vol. 1, No. 1,* pp. 3-28, 1991.

[4] D. E. McCumber, “Effect of ac Impedance on dc Voltage-Current Characteristics of Superconductor Weak-Link Junctions,” *Journal of Applied Physics, vol. 39, no. 7,* p. 3113-3118, 1968.

[5] R. Rafique, Towards Superconducting Monolithic Microwave Integrated Circuits, Chalmers University of Technology, 2008.

[6] xeoLabs, “xeogl - WebGL-based 3D visualization engine,” [オンライン]. Available: https://xeogl.org/. [アクセス日: 22 Aug. 2022].

[7] Khronos Group, “WebGL Overview - The Khronos Group Inc,” [オンライン]. Available: https://www.khronos.org/webgl/. [アクセス日: 22 Aug. 2022].

[8] Google Data Arts Team, “dataarts/dat.gui: Lightweight controller library for JavaScript,” [オンライン]. Available: https://github.com/dataarts/dat.gui. [アクセス日: 22 Aug. 2022].

[9] OpenJS Foundation, “jQuery,” [オンライン]. Available: https://jquery.com/. [アクセス日: 22 Aug. 2022].

[10] Internet Engineering Task Force (IETF), “RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format,” [オンライン]. Available: https://www.rfc-editor.org/rfc/rfc8259. [アクセス日: 22 Aug. 2022].