دانشگاه صنعتی شریف
دانشکده هوافضا

عنوان

GUI نرم افزار متلب برای تحلیل ترمودینامیکی، آیرودینامیکی و هندسی یک توربین محوری چند طبقه ای

ویرایش اول

نگارش
عبدالرضا طاهری

تیر 1394

صفحه

فهرست مطالب

1 مقدمه، نحوه ی کار کردن و اینترفیس کلی نرم افزار 3
1.1 Turbine_Stage_Design: گرفتن ورودی ها و انجام محاسبات 5
1.1.1 بدست آوردن نتایج در چهار حالت نامعلوم بودن پارامتر های مختلف 5
2.1.1 قابلیت محاسبه برای توربین یک طبقه و چند طبقه ای 6
3.1.1 پارامتر های ورودی توربین و ثابت های گاز 6
4.1.1 پارامتر های مربوط به هر طبقه (Stage) 6
2.1 Turbine_Results: مشاهده ی نتایج تحلیل 7
1.2.1 جدول خواص هر طبقه (Stage) 7
2.2.1 پنل تحلیل کلی یک طبقه (Stage) 7
3.2.1 پنل تغییر شماره ی طبقه (Stage) 8
4.2.1 شکل کلی طبقات مسئله 8
2 قابلیت ورژن کنترل 9
3 ساختار کلی و روش تحلیل نرم افزار 11
1.3 شیوه و الگوریتم محاسبات نرم افزار در قسمت اول 11
1.1.3 تعداد طبقات 11
2.1.3 پارامتر های معلوم/نا معلوم 12
3.1.3 نتایج آماده 12
4.1.3 الگوریتم محاسبات Multi-Stage 13
5.1.3 الگوریتم محاسبات درون هر طبقه 15
1.5.1.3 فرمول ها 15
2.5.1.3 محاسبات به کمک این فرمول ها 16
6.1.3 نمایش نتایج 16
2.3 نمایش نتایج در قسمت دوم 16
4 پیوست 1 18

1 مقدمه

در راستای این پروژه نسبت به ساخت یک نرم افزار اقدام کردیم که بتواند پارامتر های هر طبقه (Stage) یک توربین چند طبقه ای محوری را تحلیل کند و در طراحی توربین موثر واقع شـود. در فصـل دوم این گزارش به توضـیح شـکل کلی و نحوه ی کار با این نرم افزار می پردازیم به گونه ای که کاربر با خواندن آن سیستم را بصورت کلی درک کرده بتواند محاسبات مورد نیاز خود را انجام دهد. این نرم افزار بصورت توابع جدا در ارتباط با یکدیگر بگونه ای طراحی شده ا ست که در صورت نیاز به ایجاد تغییرات در بخش های آن، بتوان آن را در زمان کوتاه و بدون ایجاد مشکل در کارکرد بقیه قسمت ها ویرایش کرد، پس در فصل های بعدی این گزارش بصورت کامل چگونگی کار و ساختار نرم افزار را تحلیل می کنیم بگونه ای که بتوان برنامه را در آینده بهبود بخشید و قابلیت های جدید تر به آن اضافه کرد.

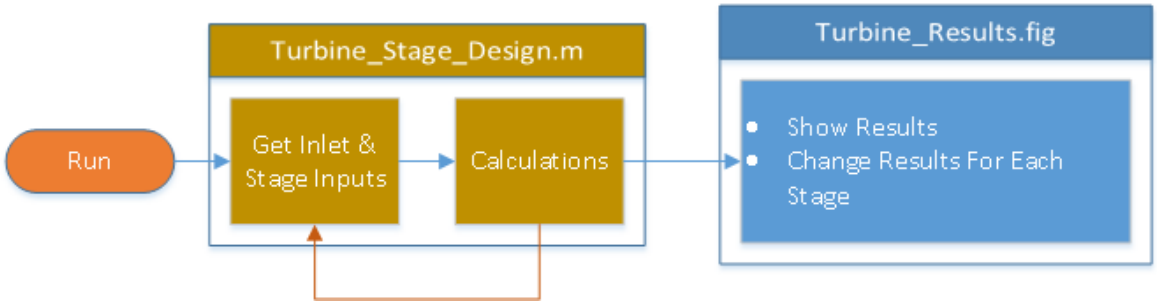
این نرم افزار به نوعی همانند نرم افزار TurbN که در کتاب (Mattingly 2006) به آن اشـاره شـده نتایج را ارائه می دهد ولی نرم افزار TurbN (حتی برای خرید) در وب یافت نمی شود و د ست عده ای خاص ا ست. از طرفی، چون این نرم افزار بصورت کد متلب ارائه شده است میتوان به راحتی قابلیت های جدید به آن ا ضافه کرد و آن را با مقالات و مدل های جدید تر به روز ر سانی کرد و آن را گ سترش داد، این در حالیست که نرم افزار هایی که در رابطه به طراحی خریداری می شوند کامپایل شده اند و سورس آن محفوظ برای شـرکت طراح می ماند و کاربر اجازه ی تغییر در الگوریتم های آن ندارد، این ها پس از مدتی و با پیدایش روش های دقیق تر، انعطاف خود را از دست می دهند. فصل چهارم به توضیح نحوه‌ی عملکرد کد می پردازد تا با درک بهتر این کد بتوان آن را ویرایش کرد.

این نرم افزار همچنین تحت کنترل ورژن طراحی شده و می تواند به صورت جداگانه توسط افراد مختلف روی آن کار شود، و در نهایت کد ویرایش شده ی همه ی افراد بدون ایجاد مشکل یکجا به نرم افزار اضافه شود. فصل سوم به این موضوع می پردازد.

ضمنا برای سادگی در توضیح الگوریتم ها و نحوه ی کارکرد، روند ها بصورت FlowChart کشیده شده اند. در صـورت هرگونه ابهام می توانید با شـماره تماس 989391935477+ و یا از طریق ایمیل Art.nt.C4@gmail.com با من تماس بگیرید.

2 نحوه ی کار کردن و اینترفیس کلی نرم افزار

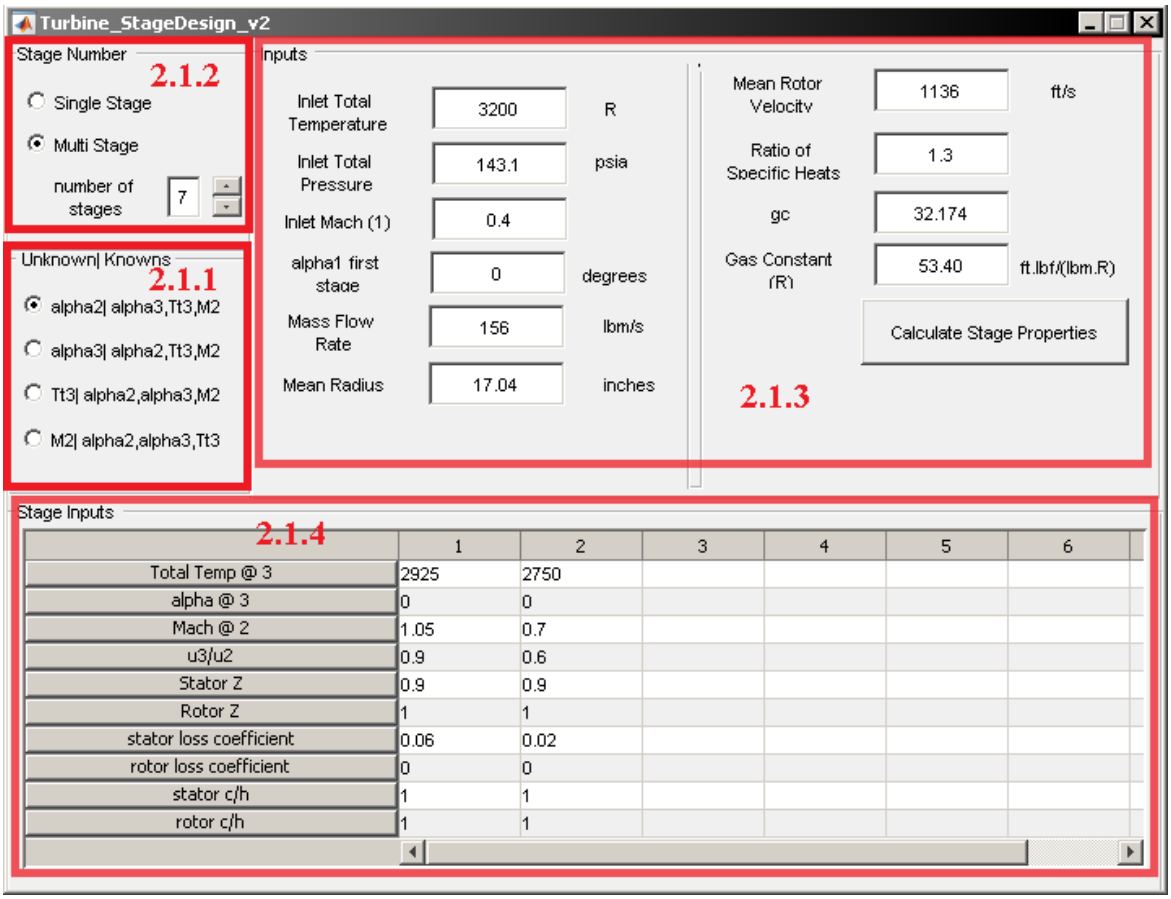
برای سادگی در طراحی، این نرم افزار به دو بخش کلی تقسیم شده است: یک بخش برای گرفتن ورودی ها و انجام محاسبات به اسم Turbine_Stage_Design و بخش دوم برای نشان دادن نتایج همه ی قسمت های توربین به نام Turbine_Results که این دو بخش در ارتباط با هم به طورت زیر کار می کنند:



در ادامه ی این فصل این دو قسمت را به صورت جداگانه بررسی می کنیم. نحوه ی کار کردن با این دو قسمت برای کاربر در ادامه آورده می شود و در فصل 4، به توضیح در مورد نحوه ی محاسبه و الگوریتم های به کار گرفته شده در نرم افزار پرداخته می شود.

1.2 Turbine_Stage_Design: گرفتن ورودی ها و انجام محاسبات

برای بدســـت آوردن پارامتر های جریان در هر طبقه (Stage)، باید ویژگی های ورودی توربین و پارامتر های هر طبقه به نرم افزار به عنوان ورودی داده شوند. شمای کلی این نرم افزار در شکل زیر آمده که در ادامه به بررسی قسمت های مختلف آن می پردازیم:



1.1.2 بدست آوردن نتایج در چهار حالت نامعلوم بودن پارامتر های مختلف

نرم افزار به گونه ای طراحی شـــده که با تغییر دکمه ی پنل (Unknown|Knowns) جداول مربوط به گرفتن پارامتر های ورودی نسبت به پارامتر های مشخص و نامشخص تغییر کنند و ورودی های متفاوت بگیرند. به عبارت دیگر، با دانســـتن ســـه پارامتر از چهار پارامتر ($M_2, \alpha_2, \alpha_3, T_{03}$) برای هر طبقه به صورت جداگانه، می‌توان به نتایج توربین محوری دست یافت.

2.1.2 قابلیت محاسبه برای توربین یک طبقه و چند طبقه ای

برای تحلیل یک طبقه از گزینه ی Single Stage و برای تحلیل یک توربین چند طبقه ای از گزینه ی Multi Stage استفاده می کنیم و سپس تعداد طبقات را تنظیم می‌کنیم. با تغییر دادن تعداد طبقات به صورت خودکار جدول مربوط به پارامتر های ورودی طبقات گسترش می یابند.

3.1.2 پارامتر های ورودی توربین و ثابت های گاز

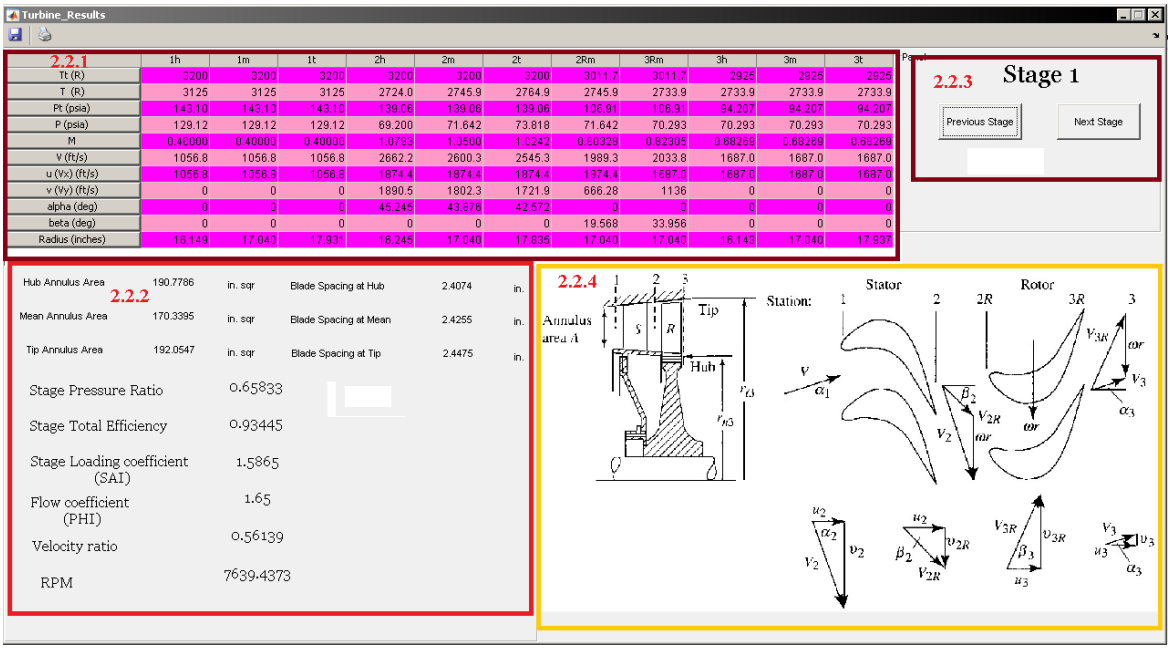
در این قسمت پارامتر های ورودی به اولین طبقه ی توربین و ثابت های گاز را وارد می کنیم، برای راحتی، ورودی های یک آزمایش مشخص در خانه های این بخش و در جدول طبقات از قبل قرار داده شده است.

4.1.2 پارامتر های مربوط به هر طبقه (Stage)

در این قسمت بسته به توع انتخاب قسمت (2.1.2) و(2.1.1) ورودی های جدول عوض می شوند و پارامتر های مختلفی برای هر طبقه باید وارد شوند. نحوه ی تحلیل نرم افزار با تغییر دکمه های رادیویی عوض می شود که در فصل چهار در مورد نحوه ی کارکرد آن توضیح داده می شود.

2.2 Turbine_Results: مشاهده ی نتایج تحلیل

این فیگور از قبل به گونه ای آماده شده تا نتایج محا سبات را از قسمت قبل بگیرد و در جدول ها و پنل های خود نشان دهد. پس از وارد کردن پارامتر ها در قسمت اول و فشردن دکمه ی Calculate، شکل زیر نمایش داده می شود:



که در زیر بخش های بعدی هرکدام از بخش های آن بررسی می شوند.

1.2.2 جدول خواص هر طبقه (Stage)

این جدول حاوی پارامتر های بدست آمده در نقاط 1، 2 و 3 ی هر طبقه از توربین است که هر یک از این نقاط در سه بخش Hub, Mean و Tip پره ها محاسبه شده اند (به شکل 2.2.4 نگاه کنید). به عنوان مثال T_{2m} نشان دهنده ی دما در خط وسط (Mean Line) نقطه ی دوم در انتهای ا ستاتور می باشد.

2.2.2 پنل تحلیل کلی یک طبقه (Stage)

در این قسمت اطلاعات کلی یک طبقه مانند نسبت فشار، بازدهی کل، ψ و ϕ، دور موتور و نسبت سرعت نشان داده می شوند.همچنین Annulus Area و فا صله ی بین پره ها در سه قسمت Tip، Mean و Hub هم نشان داده می شوند.

3.2.2 پنل تغییر شماره ی طبقه (Stage)

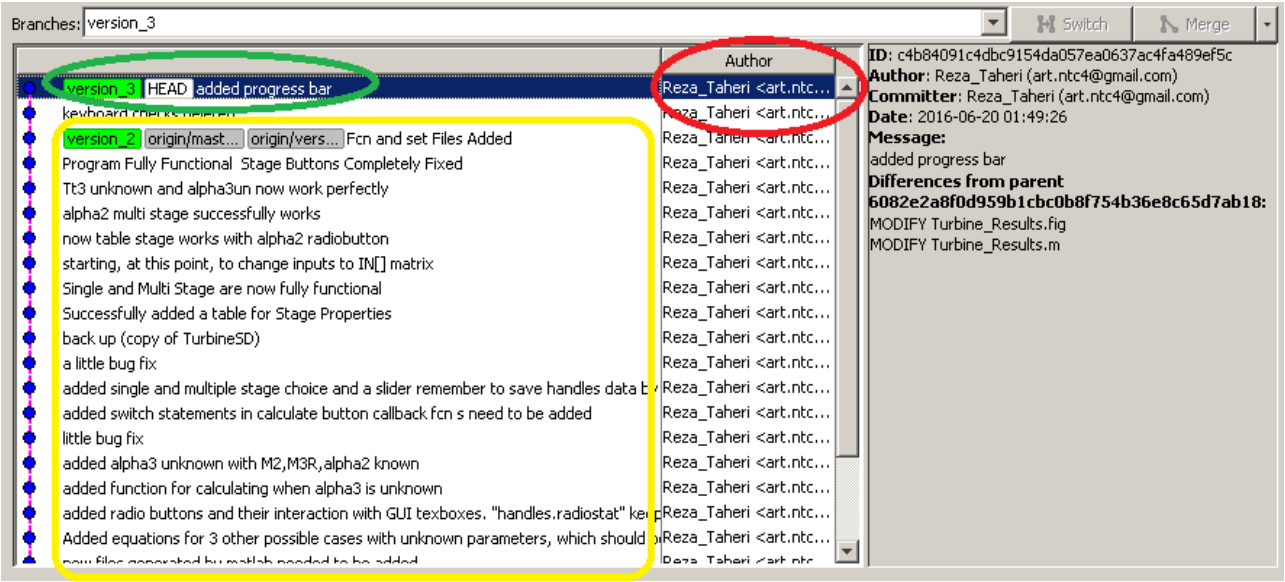
این قسمت شماره ی طبقه ی در حال نمایش در جدول 2.2.1 و پنل 2.2.2 را نشان می دهد و به کمک دو دکمه ی بعدی و قبلی می توانیم نتایج طبقه ی بعدی یا قبلی را مشاهده کنیم.

4.2.2 شکل کلی طبقات مسئله

در اینجا شکل پره های یک طبقه از دو سمت قرار داده شده تا اعداد خوانده شده در این فیگور در نقاط مختلف قابل درک باشند.

3 قابلیت ورژن کنترل

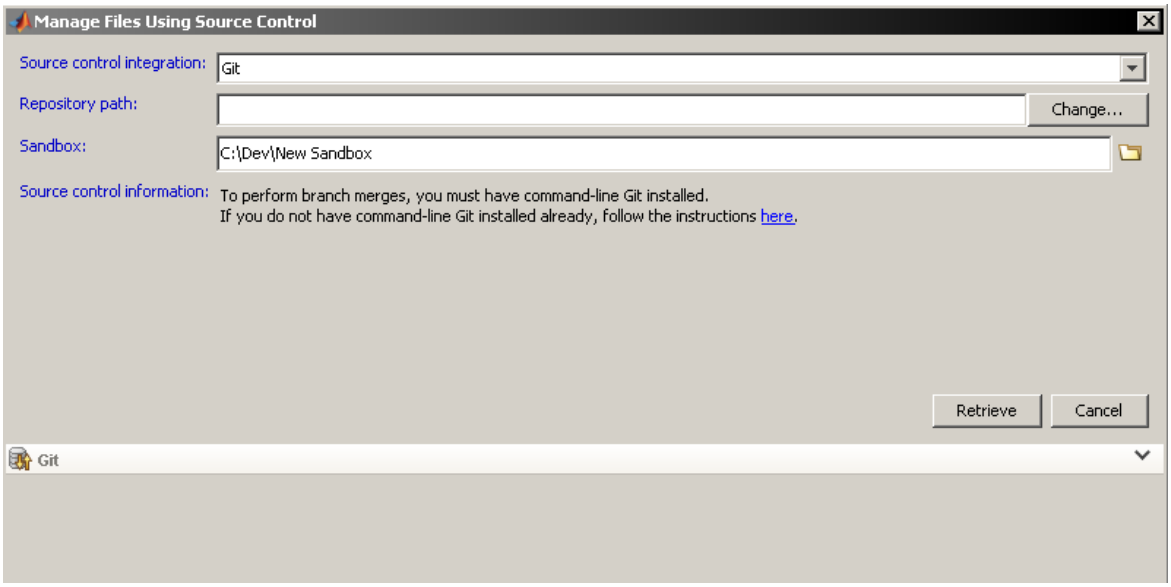
این نرم افزار از ورژن اول خود تحت کنترل ورژن گیت (Git Version Control) قرار گرفت. مزیتی که در این ویژگی وجود دارد قابلیت کار گروهی روی این پروژه است، یعنی افراد به صورت جداگانه می‌توانند روی ق سمت های متفاوتی از این نرم افزار کار کنند و سپس تمام ویرایش ها را به صورت یکجا بدون از دست رفتن اطلاعات به برنامه اعمال کنند. از طرف دیگر، در صورت خرابی و مشکل در سیستم می‌توان به راحتی و با چند کلیک ویرایش ها را برگرداند و یا به ورژن های قبلی برگشت. شکل زیر نشان دهنده‌ی اطلاعات ویرایش های اخیر این نرم افزار است. اسم فرد ویرایش کننده نیز در ادامه‌ی هر ویرایش آورده می‌شود. این پروژه فعلا به صورت یک نفره انجام می شود و یک ویرایشگر دارد:



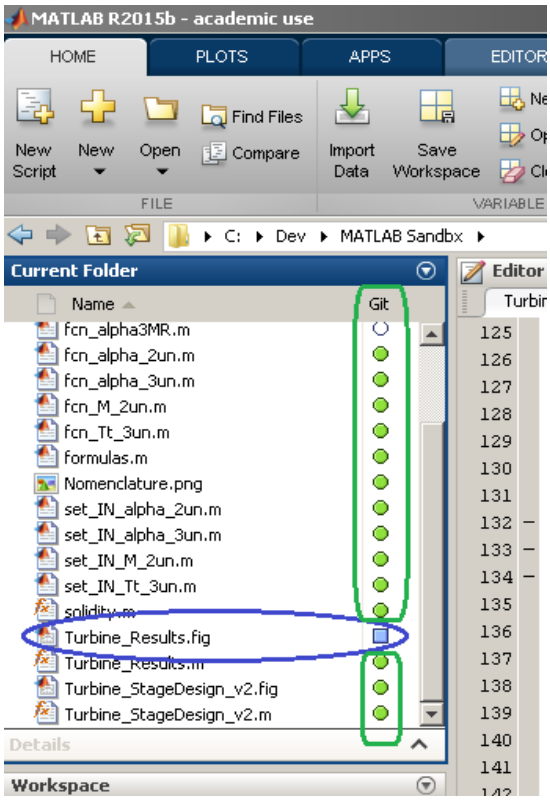
برای اســـتفاده از قابلیت ورژن کنترل باید گیت روی کامپیوتر نصـــب شـــود، با اجرای فرمان git! در Command Window متلب می توان نصب بودن گیت را چک کرد، و در صورتی که گیت نصب نباشد باید جدیدترین نسخه ی آن را از سایت http://msysgit.github.io دریافت نمود.

پس از نصـب گیت، با کلیک راسـت درون یک فولدر خالی در متلب، از منوی Source Control گزینه Manage Files را انتخـاب کنیـد. برای Source Control Integration گزینـه Git و برای Repository، فولدر Repository این پروژه را انتخاب کنید و یک فولدر خالی برای Sandbox انتخاب کنید، Sandbox یک کپی از پروژه در محیط کاری ایزوله ای اســت که در آن پروژه را ویرایش

می کنیم و تنها پس از نهایی شدن تغییرات آن وارد پروژه می کنیم تا از خراب شدن نرم افزار جلوگیری شود.



پس از انجام ویرایش روی فایل های این نرم افزار، دایره ی سبز روبروی فایل که قبلا به معنی آپدیت بودن نرم افزار بود اکنون به شکل مربع آبی در می آید، که با کلیک راست، گزینه ی Commit All از منوی Source Control همه ی این تغییرات ثبت می شوند و دوباره همه ی فایل ها آپدیت و سبز می‌شوند:



4 ساختار کلی و روش تحلیل نرم افزار

در این فصل نحوه ی کار نرم افزار به صورت جزئی تر بررسی می شود تا توسعه و اضافه کردن ویژگی های جدید به آن در صورت نیاز میسر گردد. این فصل تنها به تحلیل کد برای ویرایش آن توسط یک برنامه نویس می پردازد و در نظر داشته باشید که توضیحات فصل 2 برای استفاده کاربر از نرم افزار کافی است.

1.4 شیوه و الگوریتم محاسبات نرم افزار در قسمت اول

در این قسمت به بررسی کد هر قسمت از نرم افزار که در فصل 2 توضیح داده شد می پردازیم، با توجه به اینکه مطالعه‌ی کد باعث درک بهتر نرم افزار برای بهبود هر قسمت آن در آینده ضروری است.

در طراحی کد GUI در متلب، ساختار handles یک متغیر است که بسیاری از مقادیر نرم افزار را درون خود ذخیره می کند و در تمام توابع نرم افزار قابل دسترسی است. به همین دلیل، پارامتر های ثابت از قبیل تعداد طبقه و داده های قبلی در این متغیر ذخیره می شوند.

1.1.4 تعداد طبقات

در تحلیل توربین، با توجه به ورودی تعداد طبقات، جدول وارد کردن پارامتر های هر طبقه نسبت به کم یا زیاد کردن تعداد طبقات، گسترش یا کاهش می یابد تا ورودی ها را به تعداد درست بگیرد. کد زیر در چند قسمت به همین منظور استفاده شده است:

```
handles.stagenumber=
str2double(get(handles.stagenumber_textbox,'String'));
switch(handles.radiostat)
    case 'alpha2'
        new_data = handles.Default_Data_alpha2;
    case 'M2'
        new_data = handles.Default_Data_M2;
    case 'Tt3'
        new_data = handles.Default_Data_Tt3;
    case 'alpha3'
        new_data = handles.Default_Data_alpha3;
end
for i=3:handles.stagenumber
    new_data = [new_data,handles.Empty_Data];
end
```

```
set(handles.Table_Stage,'Data',new_data);
guidata(hObject, handles);
```

2.1.4 پارامتر های معلوم/نا معلوم

با توجه به پارامتر های معلوم و نامعلوم مسئله همانطور که در فصل 2 توضیح داده شد ورودی سطر های جدول هم تغییر می کند، اسم سطر های چهار حالت در handles.rowname_* قرار دارد که در صورت نیاز به اضافه کردن حالت دیگری به چهار حالت کنونی، باید اسم ستون های جدید اضافه شود. کد زیر اسامی ستون ها و کد داخل یک دکمه ی معلوم/نامعلوم را نشان می دهد.

```
%Default Data
handles.rowname_alpha2 = {'Total Temp @ 3','alpha @ 3','Mach @
2','u3/u2','Stator Z','Rotor Z','stator loss coefficient','rotor
loss coefficient','stator c/h','rotor c/h'};
handles.rowname_alpha3 = {'Total Temp @ 3','alpha @ 2','Mach @
2','u3/u2','Stator Z','Rotor Z','stator loss coefficient','rotor
loss coefficient','stator c/h','rotor c/h'};
handles.rowname_M2 = {'alpha @ 2','alpha @ 3','Total Temp @
3','u3/u2','Stator Z','Rotor Z','stator loss coefficient','rotor
loss coefficient','stator c/h','rotor c/h'};
handles.rowname_Tt3 = {'alpha @ 2','alpha @ 3','Mach @
2','u3/u2','Stator Z','Rotor Z','stator loss coefficient','rotor
loss coefficient','stator c/h','rotor c/h'};
%radio button
handles.radiostat = 'alpha2';
%change table_stage
set(handles.Table_Stage,'RowName',handles.rowname_alpha2);
if(get(handles.radiobutton7,'Value'))
    new_data = handles.Empty_Data;
elseif(get(handles.radiobutton8,'Value'))
    new_data = handles.Default_Data_alpha2;
    for i=3:handles.stagenumber
        new_data = [new_data,handles.Empty_Data];
    end
end
set(handles.Table_Stage,'Data',new_data);
```

3.1.4 نتایج آماده

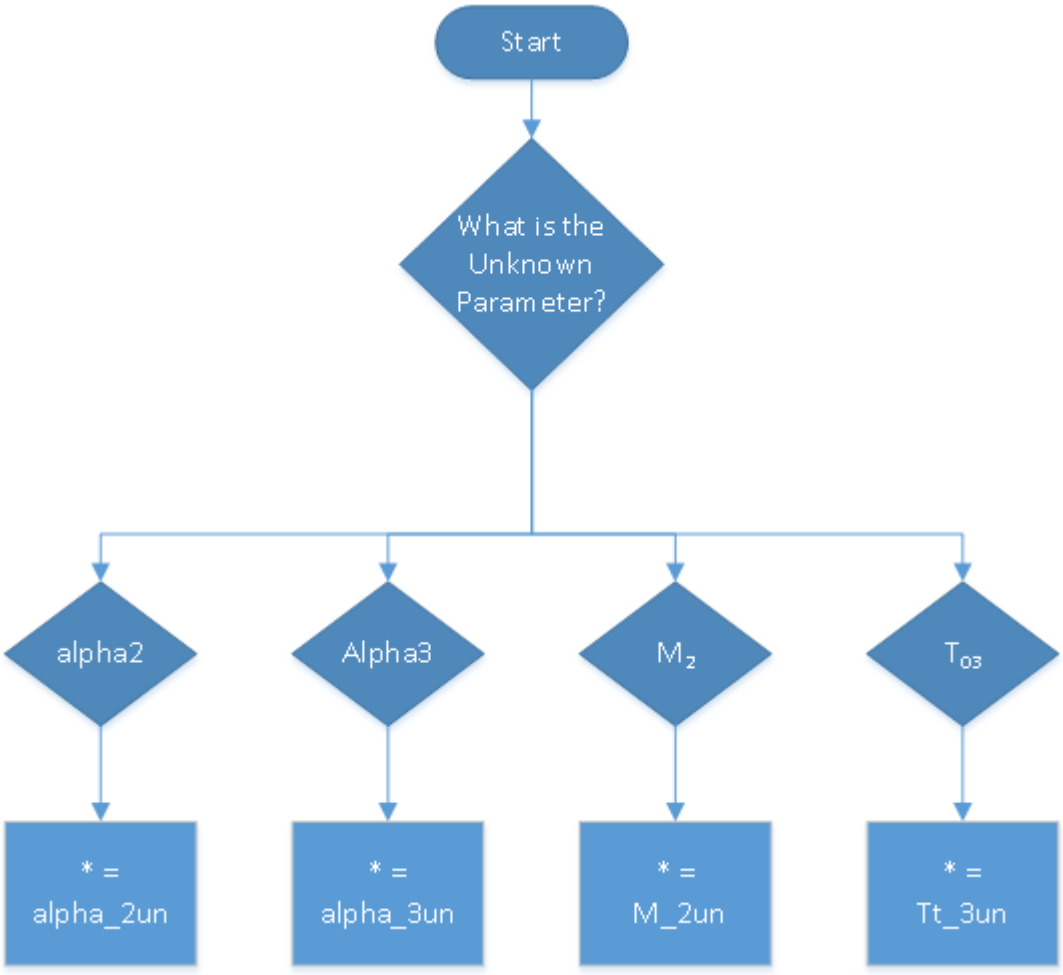
نتایج از قبل آماده شده ی مرجع 1 در این نرم افزار قرار داده شده اند تا با یک کلیک بتوان آن را تست کرد، برای انجام تغییرات یا اضافه کردن داده ها این خطوط از کد را به دلخواه عوض کنید:

```
handles.Default_Data_alpha2 =
{'2925','2750';'0','0';'1.05','0.7';'0.9','0.6';'0.9','0.9';'1','1';'0
.06','0.02';'0','0';'1','1';'1','1'};
handles.Default_Data_alpha3 =
{'2925','2750';'43.88','41.65';'1.05','0.7';'0.9','0.6';'0.9','0.9';'1
','1';'0.06','0.02';'0','0';'1','1';'1','1'};
```

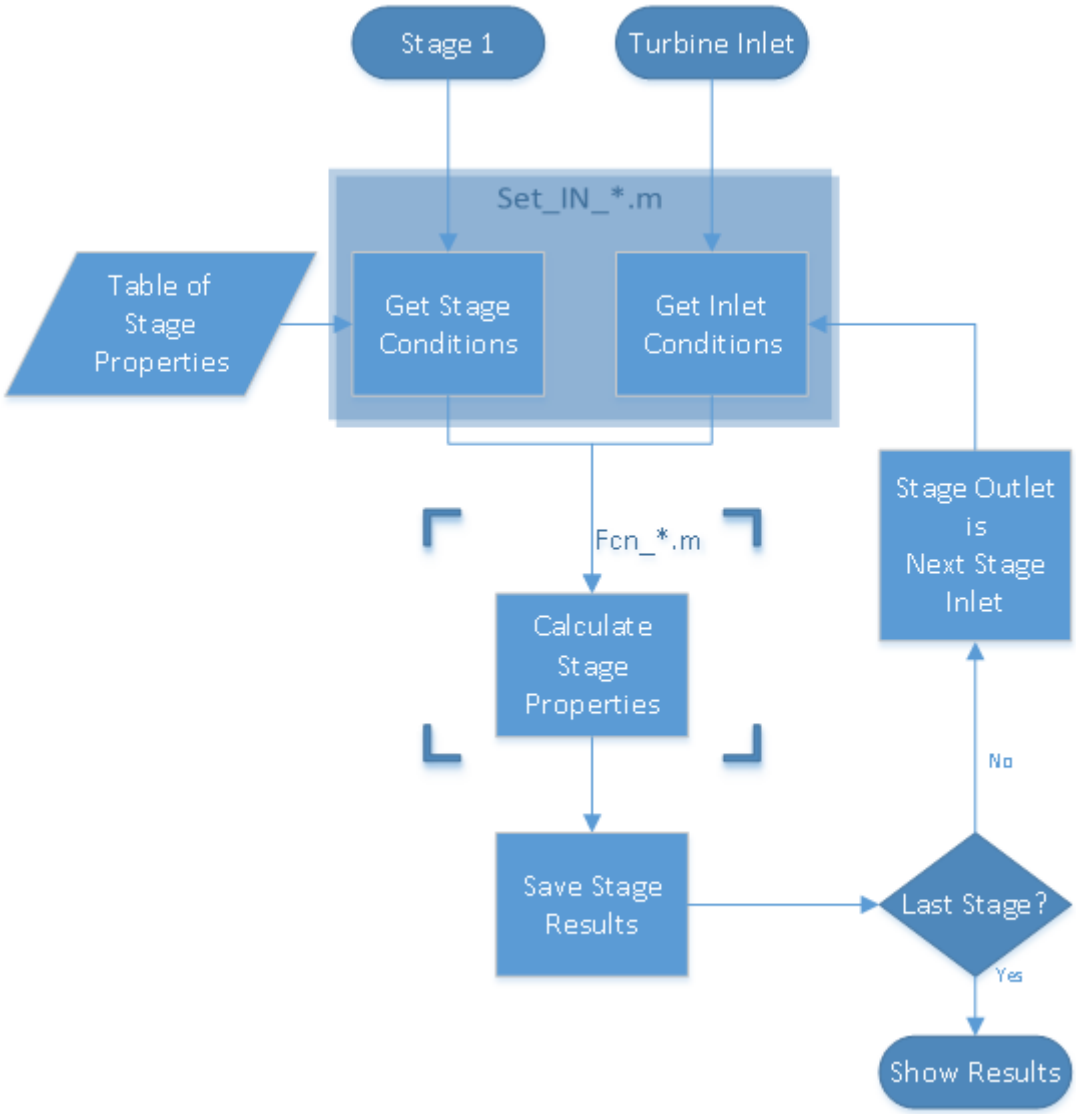
```
handles.Default_Data_M2 =
{'43.88','41.65';'0','0';'2925','2750';'0.9','0.6';'0.9','0.9';'1','1'
;'0.06','0.02';'0','0';'1','1';'1','1'};
handles.Default_Data_Tt3
={'43.88','41.65';'0','0';'1.05','0.7';'0.9','0.6';'0.9','0.9';'1','1'
;'0.06','0.02';'0','0';'1','1';'1','1'};
%add more data
```

4.1.4 الگوریتم محاسبات Multi-Stage

روند محا سبه ای که نرم افزار در پیش می گیرد به این صورت ا ست: ابتدا با د ستور سویچ نوع پارامتر نامعلوم را می گیرد و توابع مربوط به تحلیل در این حالت را اجرا می کند، یعنی در شکل زیر، نوع ستاره را مشخص می کند که در قسمت بعدی از توابع مناسب استفاده کند:



سپس از ورودی های طبقه ی اول شروع می کند و با انجام محاسبات توسط تابع های set_IN_*.m و fcn_*.m (که ستاره از قسمت قبل بدست آمده) خروجی را می گیرد و از خروجی این طبقه، ورودی طبقه ی بعد را بدست می آورد. سپس با ورودی های طبقه ی بعد که در جدول وارد شده است، ذخیره می شود تا در مرحله ی بعد اجرا شود. مقادیر همه ی مرحله ها همزمان در یک ساختار جمع آوری می شوند تا به بخش نمایش (2.2) بعدا فرستاده شوند. سپس این حلقه به تعداد طبقات تکرار می شود. الگوریتم محاسبه به صورت کد و فلوچارت در زیر نشان داده شده است و توابع محاسبات نیز بعدا ارائه می شوند:



```
switch handles.radiostat
    %% alpha2 unknown
    case 'alpha2'
        for stage_i=1:handles.stagenumber

            set_IN_alpha_2un;  %getting inputs
            fcn_alpha_2un;     %calculations take place
            handles.next_stage_data = Results_Table(:,10);
        % row 10 contains  properties at mean line / check
            handles.All_Results_Table{stage_i} = Results_Table;
            handles.All_Results_Panel{stage_i} = Results_Panel;
            guidata(hObject, handles);

        end
```

5.1.4 الگوریتم محاسبات درون هر طبقه

در این قسمت به فرمول های تابع fcn_*.m که در محاسبات قسمت قبل استفاده شد می پردازیم، خروجی این توابع محاسبات، دو ماتریس یکی 11x11 و دیگری 12x1 حاوی پارامتر های طبقه است که در قسمت 4.2 استفاده می شود.

1.5.1.4 فرمول ها

بخش عمده ی این فرمول ها از (Mattingly 2006) و باقی فرمول ها با تغییرات جزئی در فرمول های اصلی یا از طریق شکل هندسی گرفته شده اند. روش افت فشار در این پروژه با استفاده از ϕ افت فشار برای استاتور و روتور است و با توجه به ساختار برنامه، اضافه کردن مدل های افت فشار دیگر بدون هیچگونه دشواری قابل انجام است.

فرمول ها برای عدم تکرار بصورت Anonymous Functions در فایل Formulas.m قرار داده شده اند و هنگام محاسبات بارگذاری می شوند. برای نمونه بخشی از این فرمول ها را در اینجا نشان می دهیم:

```
T_Tt = @(Ttx,Mx,y)(Ttx/(1+((y-1)/2)*Mx^2));
V_Tt = @(Ttx,Mx,y,gc,R)(sqrt(2*gc*(y/(y-1))*R*Ttx/(1+2/((y-
1)*Mx^2))));
Sai = @(Tt1,Tt3,wr,gc,y,R)((Tt1-Tt3)*gc*(y/(y-1))*R/(wr)^2);
```

که اولی برای بدست آوردن دمای توتال با داشتن مقادیر دما و عدد ماخ استفاده می شود، و دومی سرعت را با وارد کردن دمای سکون و عدد ماخ می‌دهد، به عنوان مثال با کد زیر:

```
T1 = T_Tt(Tt1,M1,y);
T2 = T_Tt(Tt2,M2,y);
V2 = V_Tt(Tt2,M2,y,gc,R);
```

مقادیر دما و سرعت در نقاط 1 و 2 بدست می آیند. با این توابع، خواندن کد خیلی ساده تر می شود.

2.5.1.4 محاسبات به کمک این فرمول ها

پس از فیکس شدن فرمول ها، کد fcn_*.m شروع به بدست آوردن پارامتر های همه ی نقاط می کند، از پارامتر های ورودی استاتور شروع می کند تا به انتهای روتور برسد و تمامی این مقادیر را در طول محاسبه ثبت می کند و در نهایت داخل دو ماتریس خروجی می ریزد. کد بدست آوردن پارامتر ها به عنوان نمونه در پیوست 1 آورده شده است. این کد در فایل های fcn_*.m هم قابل مشاهده است.

6.1.4 نمایش نتایج

در نهایت با اتمام حلقه های محاسبات، این نتایج به قسمت دوم نرم افزار (2.2) فرستاده می شوند تا نمایش داده شوند:

```
%% callback Results Figure
%close the current window
close(gcf);
%call up the Results Figure
Turbine_Results(handles.All_Results_Table,handles.All_Results_Panel);
```

2.4 نمایش نتایج در قسمت دوم

پس از بدست آمدن نتایج از قسمت قبل دو ورودی بزرگ به قسمت دوم فرستاده می شوند، اولی که در جدول نشان داده می شود و حاوی n ماتریس 11x11 است که n برابر تعداد طبقات توربین می باشد، و دومی که حاوی n ماتریس 12x1 برای نتایج هر طبقه ا ست و در پنل ن شان داده می شود، تعداد طبقه برای نتایج بصورت جداگانه ارسال نمی شود، برای سادگی از طول ورودی اول با دستور length() بدست می‌آید:

```
handles.TABLEDATA = varargin{1};
set(handles.data_table,'data',handles.TABLEDATA{1});
handles.maxstage = length(handles.TABLEDATA);%get total number of
stages
handles.results_data = varargin{2};
set(handles.Pr_s_statictext,'String',num2str(handles.results_data_stag
e(1)));
set(handles.n_s_statictext,'String',num2str(handles.results_data_stage
(2)));
……
……
```

سپس با تغییر شماره طبقه تو سط دو دکمه ی Next و Previous، جدول و پنل با یک ماتریس جدید به روزرسانی می شوند.

```
%% Show new Results
set(handles.data_table,'data',handles.TABLEDATA{handles.stagenumber});
set(handles.stagenumber_statictext,'String',['Stage
',num2str(handles.stagenumber)]);
% "Results panel" data
handles.results_data_stage =
handles.results_data{handles.stagenumber};
```

5 پیوست 1

```
formulas;
T1 = T_Tt(Tt1,M1,y);
V1 = V_Tt(Tt1,M1,y,gc,R);
u1 = V1*cos((alpha1));
v1 = V1*sin((alpha1));
Tt2 = Tt1;
T2 = T_Tt(Tt2,M2,y);
V2 = V_Tt(Tt2,M2,y,gc,R);
sai = Sai(Tt1,Tt3,wr,gc,y,R);
VR = 1/sqrt(2*sai);
alpha2 = alpha2_(u3_u2,alpha3,sai,wr,V2); % revised
u2 = V2*cos(alpha2);
v2 = V2*sin(alpha2);
PHI = u2/wr;
V3 = u3_u2*cos(alpha2)/cos(alpha3)*V2;
v3 = V3*sin(alpha3);
R0_t = 1 - (1/(2*sai)*(V2/wr)^2*(1-
(u3_u2*cos(alpha2)/cos(alpha3))^2));
T3 = T2 - R0_t*(Tt1-Tt3);
M3 = M2*V3/V2*sqrt(T2/T3);
M2R= M2*sqrt(cos(alpha2)^2+(sin(alpha2)-wr/V2)^2);
M3R= M3*sqrt(cos(alpha3)^2+(sin(alpha3)+wr/V3)^2); % corrected
Tt3R = Tt3 + V3^2/(2*gc*(y/(y-
1))*R)*(cos(alpha3)^2+(sin(alpha3)+wr/V3)^2-1);
Tt2R = Tt3R;
ts = Tt3/Tt1;
%
P1 = P_TTt(Pt1,Tt1,T1,y);
Pt2 = Pt_phi(Pt1,Tt2,T2,y,phi_s);
P2 = P_TTt(Pt2,Tt2,T2,y);
Pt2R = P_TTt(P2,T2,Tt2R,y); %check formula
Pt3R = Pt_phi(Pt2R,Tt3R,T3,y,phi_r);
P3 = P_TTt (Pt3R,Tt3R,T3,y);
Pt3 = P_TTt(P3,T3,Tt3,y);
Pr_s = Pt3/Pt1;
n_s = (1-ts)/(1-Pr_s^((y-1)/y));

%1
A1 = Ai_(m_dot,Tt1,Pt1,alpha1,MFP(M1,y,gc,R));
h1 = A1/(2*pi*rm);
r1h = rm - h1/2;
r1t = rm + h1/2;
%
v1h = v1*rm/r1h;
a1h = atan(v1h/u1);
v1t = v1*rm/r1t;
a1t = atan(v1t/u1);
```

```
%2
A2 = Ai_(m_dot,Tt2,Pt2,alpha2,MFP(M2,y,gc,R));
h2 = A2/(2*pi*rm);
r2h = rm - h2/2;
r2t = rm + h2/2;
v2h = v2*rm/r2h;
a2h = atan(v2h/u2);
v2t = v2*rm/r2t;
a2t = atan(v2t/u2);
% Star*
c =  c_h*(h1 + h2)/2; %chord of the stator
%end of stator
cx_s_m = Zs_cx_s(alpha1,u1,alpha2,u2)/Zs;
[y2m,t_m,sol_m] = solidity(alpha1,alpha2,cx_s_m);
blade_spacing_m = c/sol_m;

cx_s_h = Zs_cx_s(a1h,u1,a2h,u2)/Zs;
[y2h,t_h,sol_h] = solidity(a1h,a2h,cx_s_h);
blade_spacing_h = c/sol_h;

cx_s_t = Zs_cx_s(a1t,u1,a2t,u2)/Zs;
[y2t,t_t,sol_t] = solidity(a1t,a2t,cx_s_t);
blade_spacing_t = c/sol_t;

%blade exit angle
% exit_angle = (yi + ye)
%
A3 = Ai_(m_dot,Tt3,Pt3,alpha3,MFP(M3,y,gc,R));
%v3 = V3*sin(alpha3); commented due to redundancy
u3 = V3*cos(alpha3);
h3 = A3/(2*pi*rm);
r3h = rm - h3/2;
r3t = rm + h3/2;
v3h = v3*rm/r3h;
a3h = atan(v3h/u3);
v3t = v3*rm/r3t;
a3t = atan(v3t/u3);

PHI_ = u2/wr;
RPM_ = wr/rm*30/pi*12;
AN_2 = A2*RPM_^2;
B2 = atan((v2-wr)/u2);
B3 = atan((v3+wr)/u3);
Results_Table(1,:) =
[Tt1,Tt1,Tt1,Tt2,Tt2,Tt2,Tt2R,Tt3R,Tt3,Tt3,Tt3];
Results_Table(3,:) =
[Pt1,Pt1,Pt1,Pt2,Pt2,Pt2,Pt2R,Pt3R,Pt3,Pt3,Pt3];
Results_Table(7,:) = [u1,u1,u1,u2,u2,u2,u2,u3,u3,u3,u3];
Results_Table(8,:) = [v1h,v1,v1t,v2h,v2,v2t,v2R,v3R,v3h,v3,v3t];
...
...
...
...
```