Altair

HyperWorks

Altair MotionView 2019 Tutorials

MV-3022: Optimization of a 4-bar Model

altairhyperworks.com

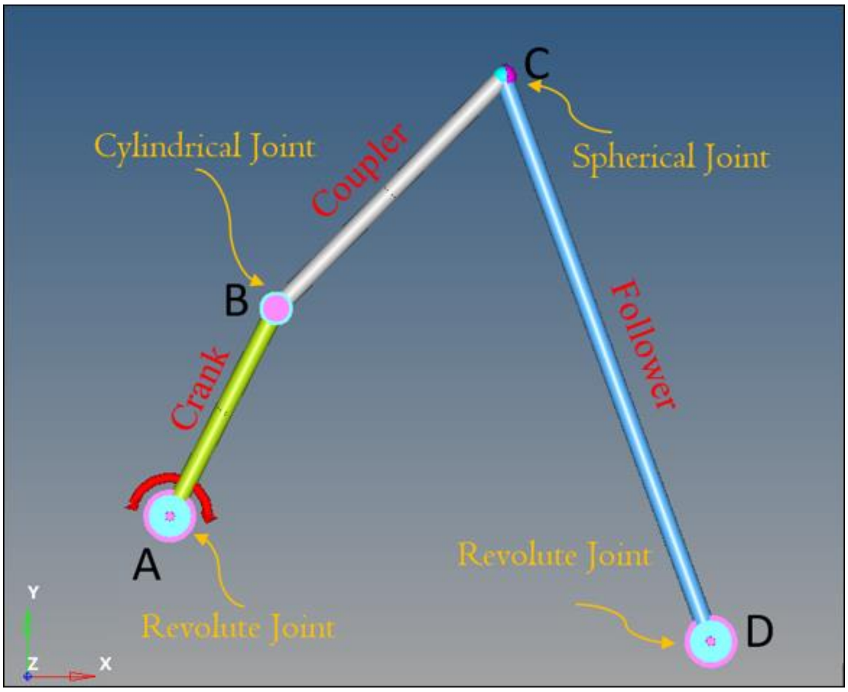
MV-3022: Optimization of a 4-bar Model

In this tutorial you will learn how to setup an optimization problem using MotionView's Optimization Wizard for a 4-bar model. You will learn about:

- Defining point coordinates as design variables
- Defining a response type 'Root Mean Square Deviation' for matching curves
- Using the responses as objectives
- Running the optimization and post-processing the results
- [Optional] Infeasible Designs in MotionSolve optimization

Introduction

In this tutorial, the locations of the joints of a 4-bar mechanism are optimized to obtain a desired motion of the coupler. MotionSolve's DSA (Design Sensitivity Analysis) capability is used to calculate sensitivities.



4 bar model with joint types

The entire model is parameterized in terms of these four design points: A, B, C and D. Since the model operates in 2D space (X-Y plane), this leads to 8 Design Variables. The initial design is listed as follows:

Point	X	Y
A	-45	45
B	65	260
C	300	500
D	515	-85

Model Setup

Copy the file `mv_3022_initial_4bar_opt.mdl` located in the `mbd_modeling\motionsolve\optimization\MV-3022` into your `<working directory>`.

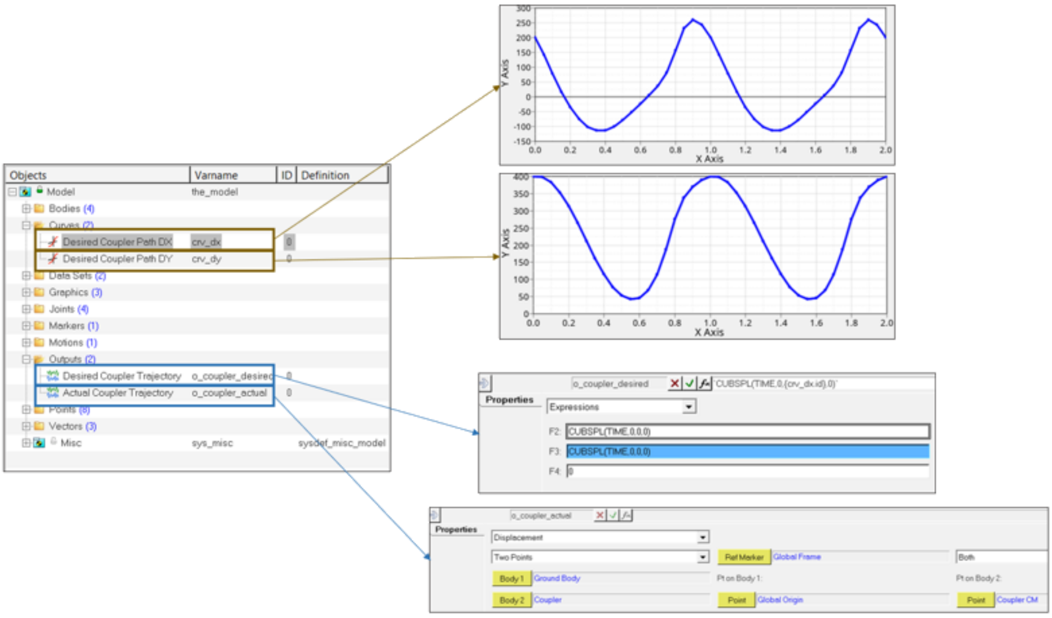
Step 1: Review the Model.

1. Open `mv_3022_initial_4bar_opt.mdl` in MotionView.
2. The model has the following set up to measure the desired motion and the actual motion of the coupler CM marker. Please review these entities in the model.

   a. Curves **Desired Coupler Path DX** and **Desired Coupler Path DY** that defines the desired motion in X and Y direction.

   b. Output **Desired Coupler Trajectory** uses the above curves to measure the desired motion.

   c. Output **Actual Coupler Trajectory** measures the displacement of the coupler body at CM.



Objects	Varname	ID	Definition
Model	the_model		
Bodies (4)			
Curves (2)			
Desired Coupler Path DX	crv_dx	0	
Desired Coupler Path DY	crv_dy	0	
Data Sets (2)			
Graphics (3)			
Joints (4)			
Markers (1)			
Motions (1)			
Outputs (2)			
Desired Coupler Trajectory	o_coupler_desired	0	
Actual Coupler Trajectory	o_coupler_actual	0	
Points (8)			
Vectors (3)			
Misc	sys_misc		sysdef_misc_model

o_coupler_desired `CUBSPL(TIME,0,{crv_dx.id},0)`

Properties: Expressions

F2: CUBSPL(TIME,0,0,0)

F3: CUBSPL(TIME,0,0,0)

F4: 0

o_coupler_actual

Properties: Displacement

Two Points — Ref Marker: Global Frame — Both

Body 1: Ground Body — Pt on Body 1: — Pt on Body 2:

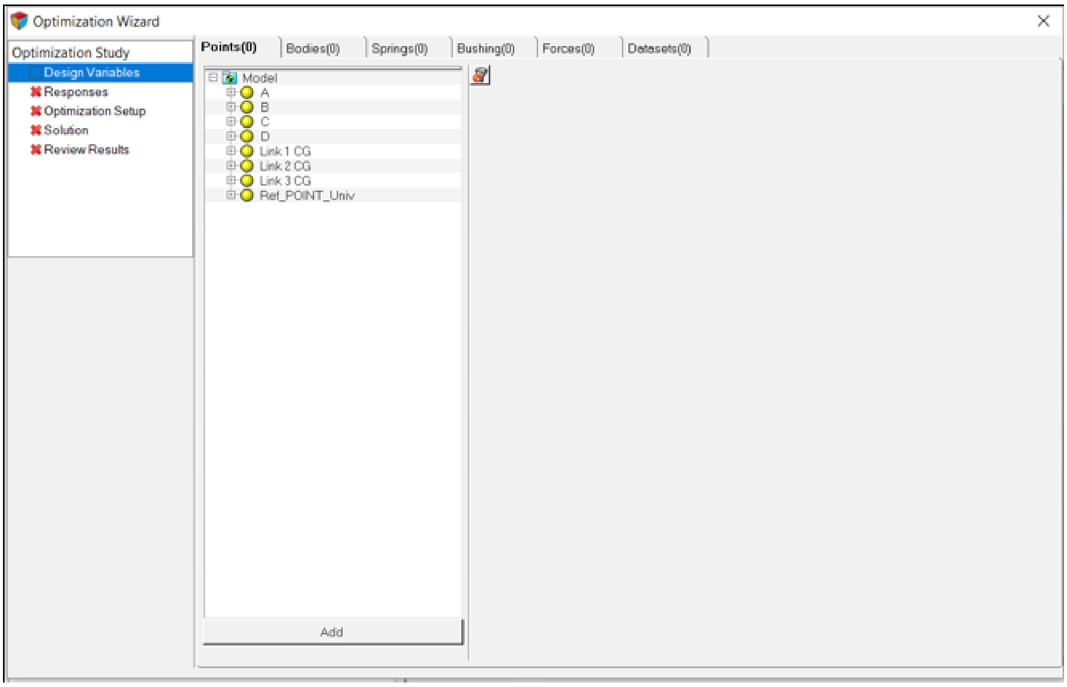
Body 2: Coupler — Point: Global Origin — Point: Coupler CM

Step 2: Adding Design Variables.

1. Right-click on the ***Model*** system in the browser and launch the **Optimization Wizard**.

2. In the **Design Variables** page, click on the ***Points*** tab.

   All points are listed as shown below:

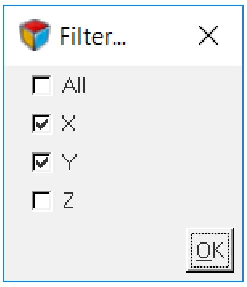


Optimization Wizard – Points Tab

3. Make the x and y coordinates of Points A, B, C and D as design variables. Click on the ***Filter datamembers*** icon above the model tree.

   a. The **Filter** dialog is displayed.

   b. Turn off the ***Z*** check box (as shown below).



□ All
☑ X
☑ Y
□ Z

   c. Click ***OK***.

   d. Select ***Point A***, ***Point B***, ***Point C*** and ***Point D*** in the model tree and click ***Add***.

4. Modify the upper and lower bounds of the design variables as listed in the table below.

DV	Lower Bound	Upper Bound
Point A(DV) - X	-50	50
Point A(DV) - Y	-50	50
Point B(DV) - X	20	80
Point B(DV) - Y	180	280
Point C(DV) - X	240	380
Point C(DV) - Y	400	620
Point D(DV) - X	180	520
Point D(DV) - Y	-100	20

The setting of design variables is now complete.

Step 3: Adding Response Variables.

The objective of the optimization is to make the coupler move along a path. To achieve this, two responses are created:

a) Trajectory of Coupler CM – DX

b) Trajectory of Coupler CM – DY

The above curves of the coupler motion must be matched with their respective target curves using a response variable of type Root Mean Square Deviation (RMS2). The details of this response can be obtained from the *Multibody Optimization User's Guide* and the *MotionView User's Guide*.

1. Click on the ***Responses*** page.
2. Click on the ➕ button.
3. Change the Label to ***'rms2_dx'***.
4. Click ***OK*** to create a response variable.
5. Once the response variable is created, under **Response Type**, choose ***Root Mean Square Deviation***.

   This response needs two user inputs:

   - Desired Curve – This is the target curve.
   - Response Expression – This is the value we measure (Coupler cm motion).

6. Double click on the ***Curve*** collector and choose ***Desired Coupler Path DX*** through the model tree for **Desired Curve**. This is the target curve for Couple-DX.

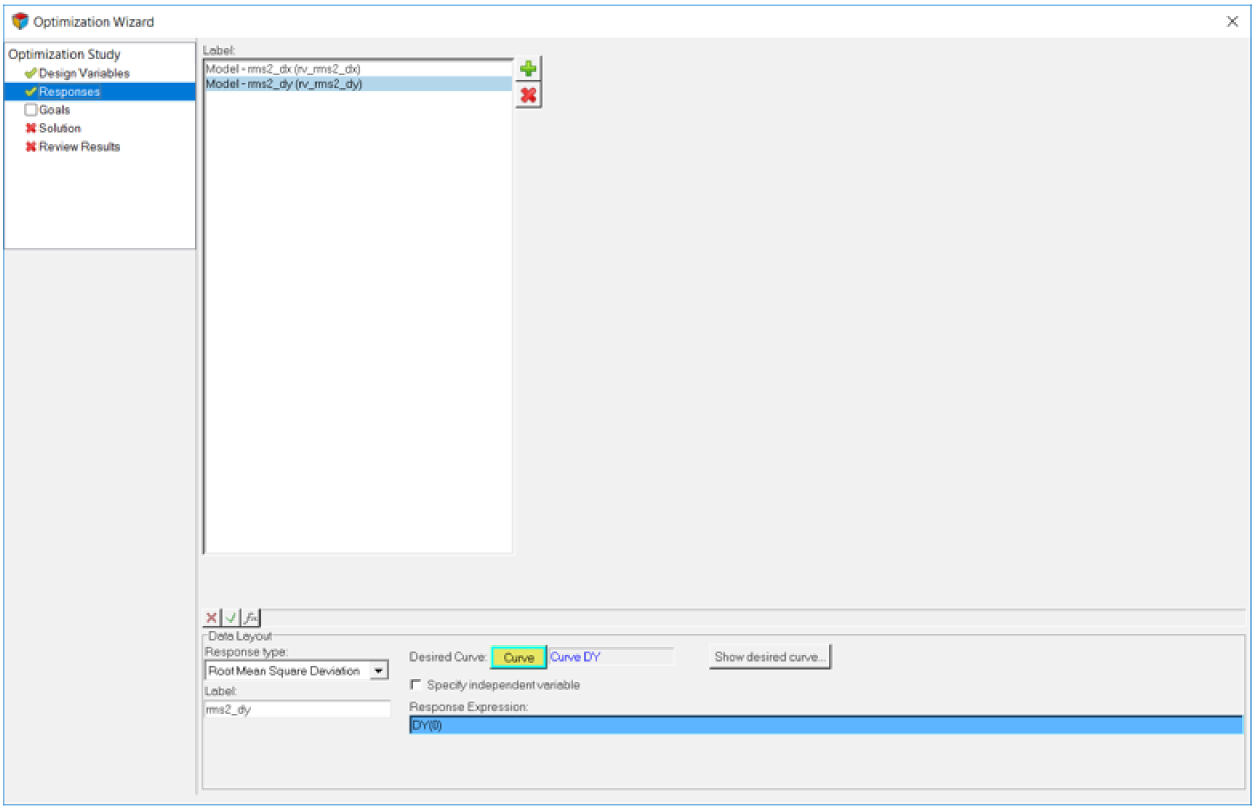
   Note The desired curve can be visualized by clicking the ***Show Desired Curve*** button.

7. For **Response Expression**, type in the expression for DX of CM of couple body: `` `DX({b_bc.cm.id})` ``.

8. Following the same procedure, add one more ***Root Mean Square Deviation*** response and fill in the details using the table below.

RV	Desired Curve	Response Expression
rms2_dx	Desired Coupler Path DX	`` `DX({b_bc.cm.id})` ``
rms2_dy	Desired Coupler Path DY	`` `DY({b_bc.cm.id})` ``

This completes the process of adding response variables; the completed **Responses** page should appear as shown below.



Label:
Model - rms2_dx (rv_rms2_dx)
Model - rms2_dy (rv_rms2_dy)

Response type: Root Mean Square Deviation

Label: rms2_dy

Desired Curve: Curve Curve DY

□ Specify independent variable

Response Expression: DY(0)

Completed Responses page

Step 4: Adding Objectives and Constraints.

The problem has two objectives: the coupler DX and DY motions should match with target curves at the end of optimization.

We can use the responses that were created in the previous section as objectives.

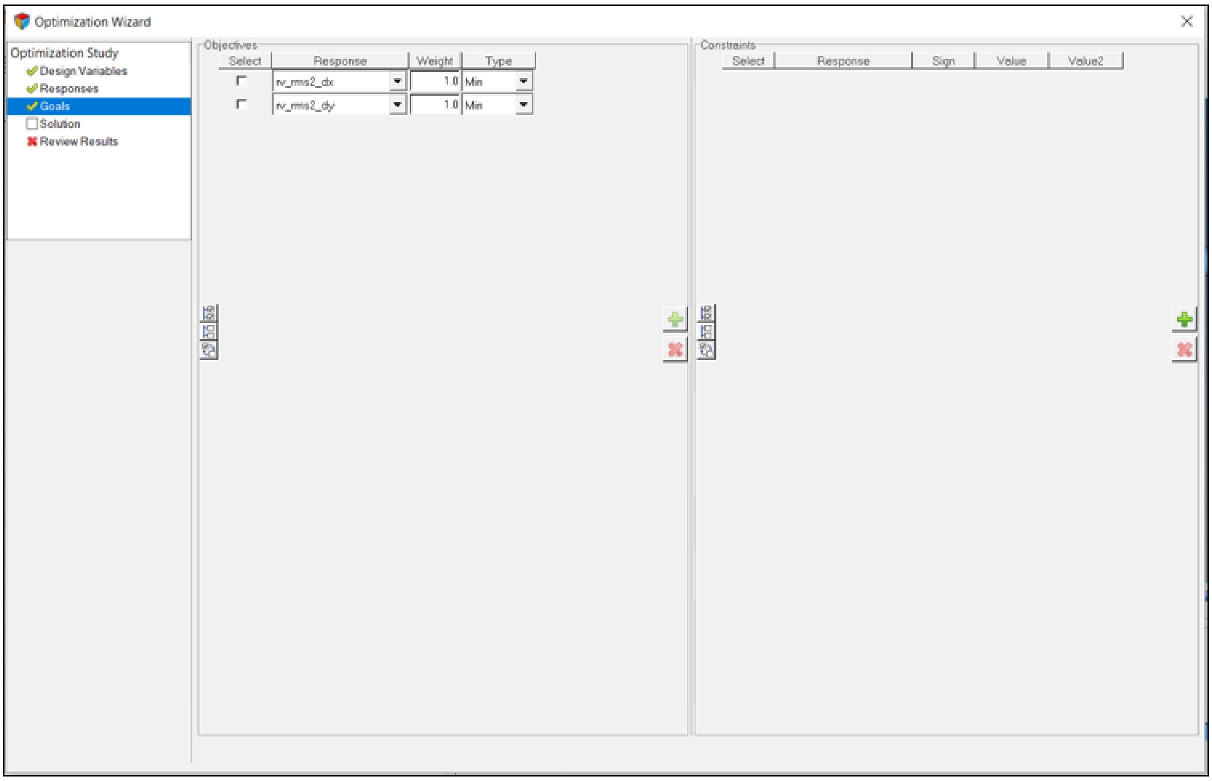
1. Go to the **Goals** page.
2. Click on the ✚ button under **Objectives**.

   An objective is added with response ***rv_rms2_dx***.
3. Choose **Weight** as `1.0` and retain the **Type** as ***min*** since we want to minimize this response.
4. Repeat the step above to add another objective. Choose ***rv_rms2_dy*** for the second objective.

   The **Weight** would be `1.0` and the **Type** would be ***min***.

There are no constraints in this problem.

The model setup is now complete and it is ready to solve.



Objectives

Select	Response	Weight	Type
□	rv_rms2_dx	1.0	Min
□	rv_rms2_dy	1.0	Min

Constraints

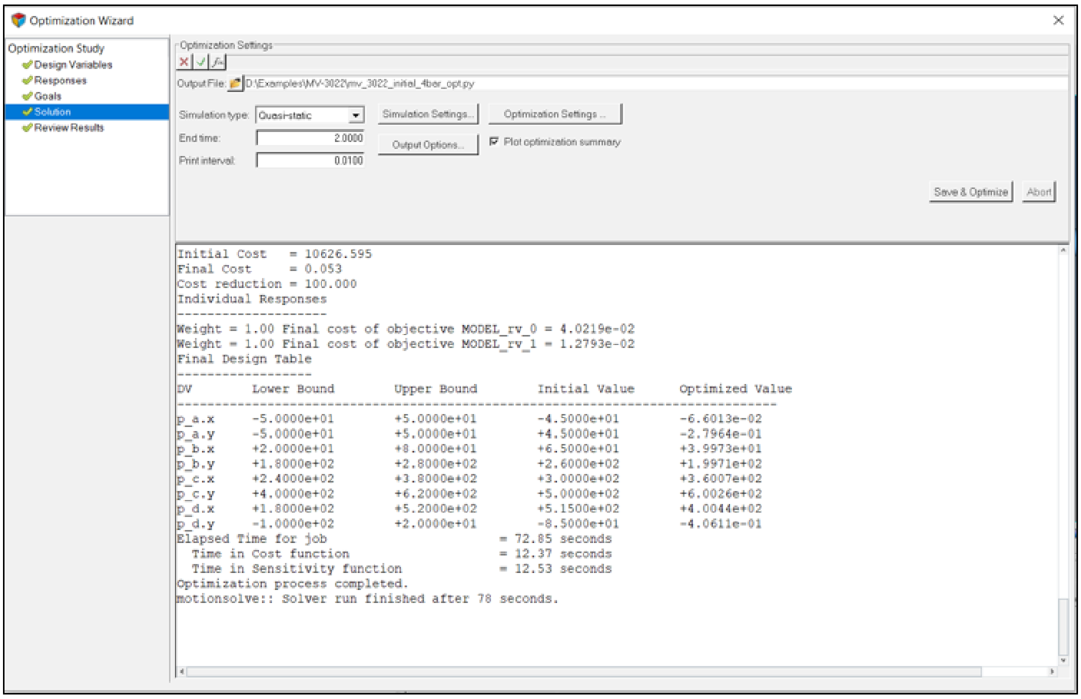
Select	Response	Sign	Value	Value2

Defining objectives

Step 5: Running the Optimization.

1. Go to the **Solution** page.
2. Activate the ***Plot Optimization Summary*** check box.
3. Click on ***Save & Optimize*** to start the optimization.

Once the optimization process is complete, the text window displays the optimized design variables values, final value of the responses and optimized cost function.



Output File: D:\Examples\MV-3022\mv_3022_initial_4bar_opt.py

Simulation type: Quasi-static

End time: 2.0000

Print interval: 0.0100

☑ Plot optimization summary

```
Initial Cost   = 10626.595
Final Cost     = 0.053
Cost reduction = 100.000
Individual Responses
--------------------
Weight = 1.00 Final cost of objective MODEL_rv_0 = 4.0219e-02
Weight = 1.00 Final cost of objective MODEL_rv_1 = 1.2793e-02
Final Design Table
------------------
DV        Lower Bound        Upper Bound        Initial Value      Optimized Value
--------------------------------------------------------------------------------------
p_a.x     -5.0000e+01        +5.0000e+01        -4.5000e+01        -6.6013e-02
p_a.y     -5.0000e+01        +5.0000e+01        +4.5000e+01        -2.7964e-01
p_b.x     +2.0000e+01        +8.0000e+01        +6.5000e+01        +3.9973e+01
p_b.y     +1.8000e+02        +2.8000e+02        +2.6000e+02        +1.9971e+02
p_c.x     +2.4000e+02        +3.8000e+02        +3.0000e+02        +3.6007e+02
p_c.y     +4.0000e+02        +6.2000e+02        +5.0000e+02        +6.0026e+02
p_d.x     +1.8000e+02        +5.2000e+02        +5.1500e+02        +4.0044e+02
p_d.y     -1.0000e+02        +2.0000e+01        -8.5000e+01        -4.0611e-01
Elapsed Time for job                          = 72.85 seconds
  Time in Cost function                       = 12.37 seconds
  Time in Sensitivity function                = 12.53 seconds
Optimization process completed.
motionsolve:: Solver run finished after 78 seconds.
```

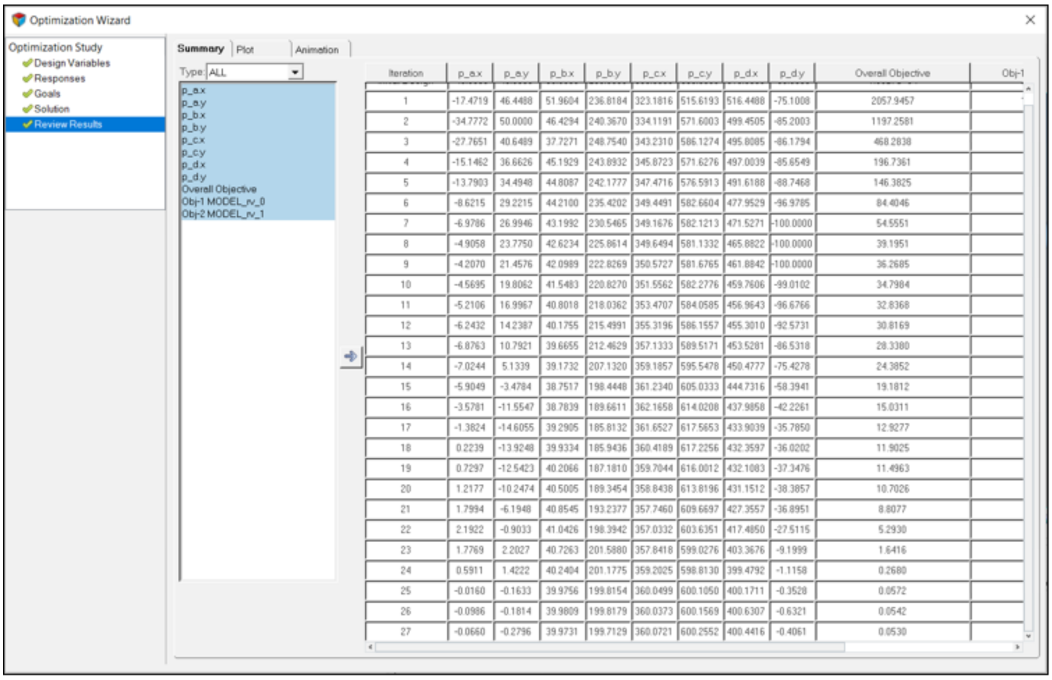
Optimization text window after the optimization is completed

The expected values of design variables are given below:

DV	Expected	From Optimizer
p_0.x	0	-0.0660
p_0.y	0	-0.2796
p_1.x	40	39.97
p_1.y	200	199.71
p_2.x	360	360.07
p_2.y	600	600.26
p_3.x	400	400.44
p_3.y	0	-0.0406

Step 6: Post-processing

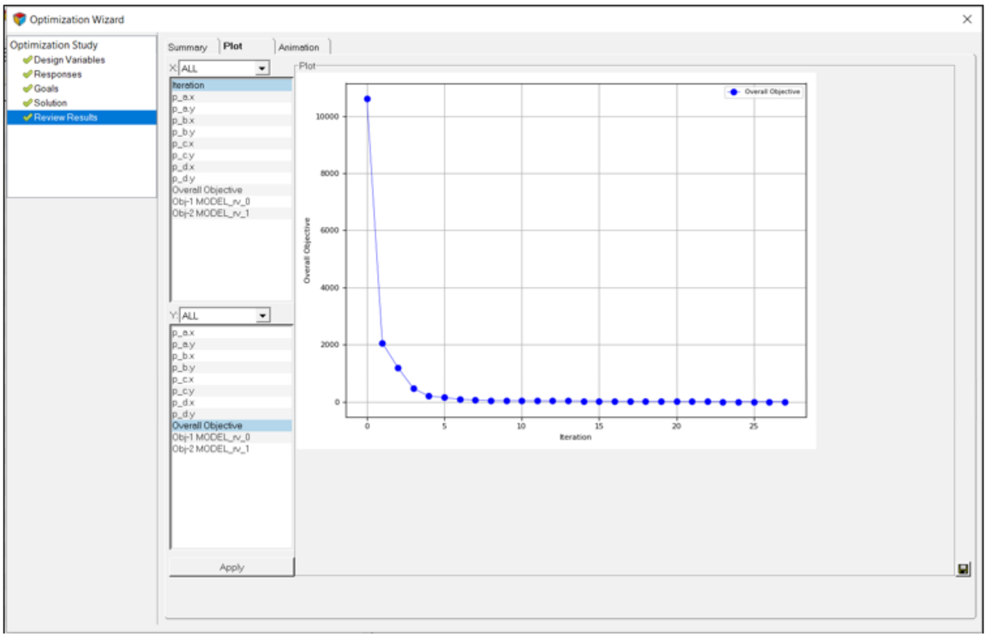
1. Go to the **Review Results** page which is enabled once the solution is finished.
2. The ***Summary*** tab lists the history of design variables, responses and objective functions in a tabular format. In this tutorial, the optimized design variables are from iteration 27.



Iteration	p_a.x	p_a.y	p_b.x	p_b.y	p_c.x	p_c.y	p_d.x	p_d.y	Overall Objective	Obj-1
1	-17.4719	46.4488	51.9604	236.8184	323.1816	515.6193	516.4488	-75.1008	2057.9457	
2	-34.7772	50.0000	46.4294	240.3670	334.1191	571.6003	499.4505	-85.2003	1197.2581	
3	-27.7651	40.6489	37.7271	248.7540	343.2310	586.1274	495.8085	-86.1794	468.2838	
4	-15.1462	36.6626	45.1929	243.8932	345.8723	571.6276	497.0039	-85.6549	196.7361	
5	-13.7903	34.4948	44.8087	242.1777	347.4716	576.5913	491.6188	-88.7468	146.3825	
6	-8.6215	29.2215	44.2100	235.4202	349.4491	582.6604	477.9529	-96.9785	84.4046	
7	-6.9786	26.9946	43.1992	230.5465	349.1676	582.1213	471.5271	-100.0000	54.5551	
8	-4.9058	23.7750	42.6234	225.8614	349.6494	581.1332	465.8822	-100.0000	39.1951	
9	-4.2070	21.4576	42.0989	222.8269	350.5727	581.6765	461.8842	-100.0000	36.2685	
10	-4.5695	19.8062	41.5483	220.8270	351.5562	582.2776	459.7606	-99.0102	34.7984	
11	-5.2106	16.9967	40.8018	218.0362	353.4707	584.0585	456.9643	-96.6766	32.8368	
12	-6.2432	14.2387	40.1755	215.4991	355.3196	586.1557	455.3010	-92.5731	30.8169	
13	-6.8763	10.7921	39.6655	212.4629	357.1333	589.5171	453.5281	-86.5318	28.3380	
14	-7.0244	5.1339	39.1732	207.1320	359.1857	595.5478	450.4777	-75.4278	24.3852	
15	-5.9049	-3.4784	38.7517	198.4448	361.2340	605.0333	444.7316	-58.3941	19.1812	
16	-3.5781	-11.5547	38.7839	189.6611	362.1658	614.0208	437.9858	-42.2261	15.0311	
17	-1.3824	-14.6055	39.2905	185.8132	361.6527	617.5653	433.9039	-35.7850	12.9277	
18	0.2239	-13.9248	39.9334	185.9436	360.4189	617.2256	432.3597	-36.0202	11.9025	
19	0.7297	-12.5423	40.2066	187.1810	359.7044	616.0012	432.1083	-37.3476	11.4963	
20	1.2177	-10.2474	40.5005	189.3454	358.8438	613.8196	431.1512	-38.3857	10.7026	
21	1.7994	-6.1948	40.8545	193.2377	357.7460	609.6697	427.3557	-36.8951	8.8077	
22	2.1922	-0.9033	41.0426	198.3942	357.0332	603.6351	417.4850	-27.5115	5.2930	
23	1.7769	2.2027	40.7263	201.5880	357.8418	599.0276	403.3676	-9.1999	1.6416	
24	0.5911	1.4222	40.2404	201.1775	359.2025	598.8130	399.4792	-1.1158	0.2680	
25	-0.0160	-0.1633	39.9756	199.8154	360.0499	600.1050	400.1711	-0.3528	0.0572	
26	-0.0986	-0.1814	39.9809	199.8179	360.0373	600.1569	400.6307	-0.6321	0.0542	
27	-0.0660	-0.2796	39.9731	199.7129	360.0721	600.2552	400.4416	-0.4061	0.0530	

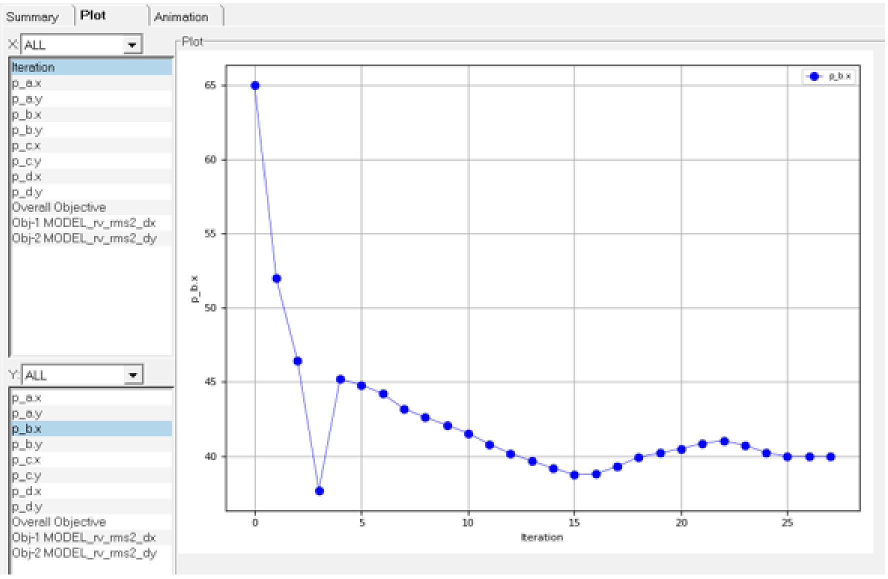
Summary tab listing design variables, responses and cost function iteration-wise

3. The **Plot** tab helps to visualize variation of design variables, response variables and cost function using graphs. You can choose to plot any number of variables along the y-axis with respect to a variable along the x-axis. For this tutorial, choose ***Overall Objective*** vs ***Iteration*** and you should get a plot as shown below:



Plot of overall objective (log scale) vs. iteration number.

4. You can also see how a particular design variable varied across an iteration. For example, select ***p_b.x***(Point B-X) in the **Y** list and click ***Apply***.



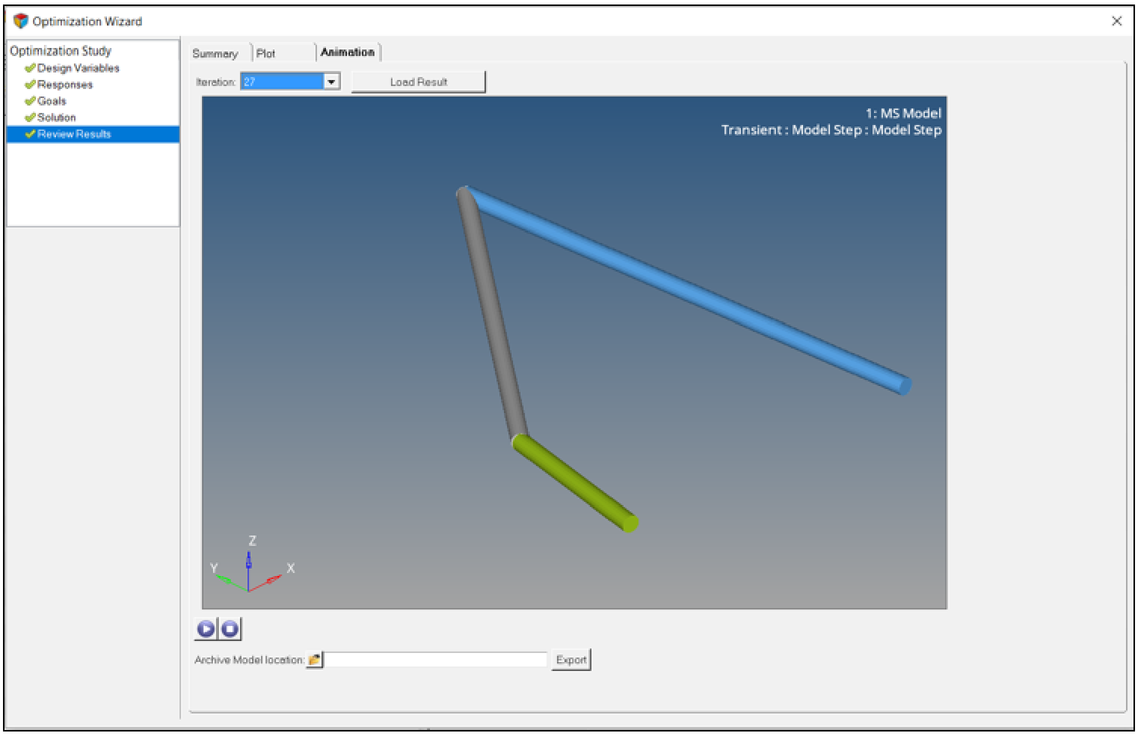
5. The **Animation** tab helps you to animate the configuration generated during any iteration.

   a. Choose the iteration number from the ***Iteration*** drop-down menu.

   b. Click on ***Load Result*** to load the animation corresponding to that iteration.

   c. Start, pause and stop animation buttons are provided at the bottom.

   For this problem, load the animation file from the last iteration (iteration 27) and animate the configuration.



Animation tab in the Post Processing page

6. The **Export** option is available in this tab for you to export an archive of the model from any configuration.

   For this tutorial, we would want to create an mdl model for the optimized configuration.

   a. Load results from iteration 27.

   b. Choose a folder at **Archive Model location**.

   c. Click on ***Export***.

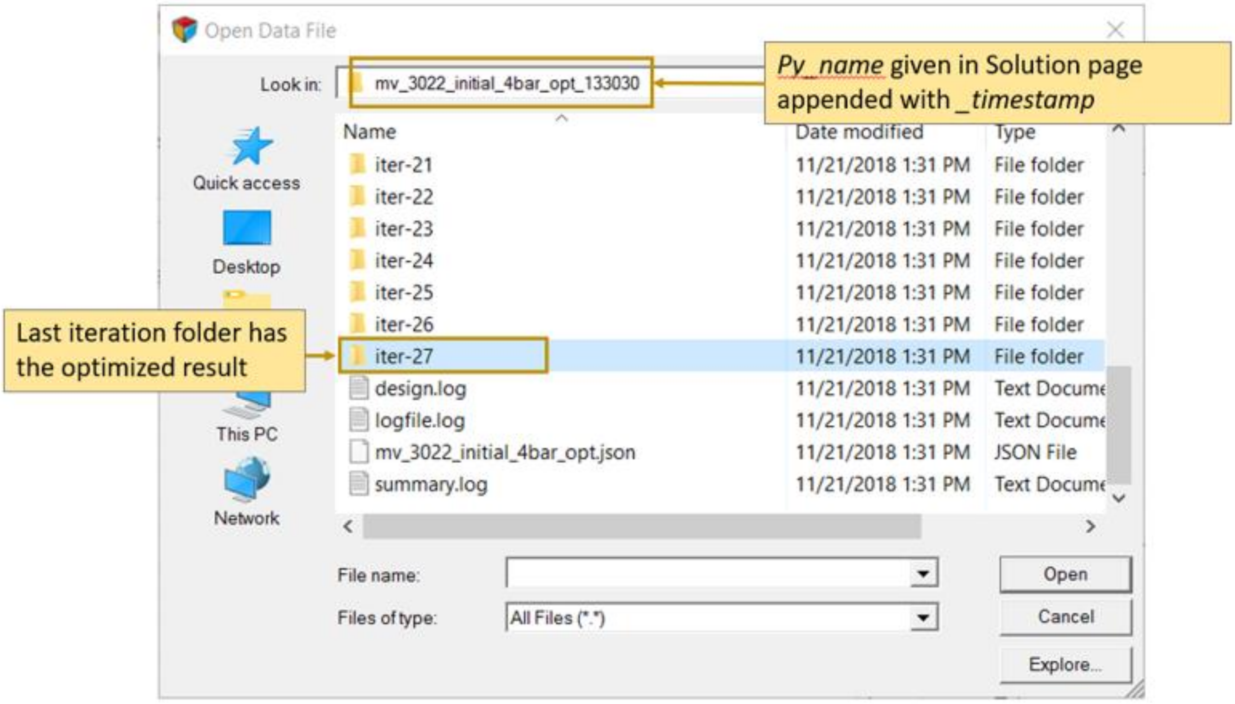
7. Finally, we will look at how the path of the coupler for the optimized result compare with the desired target path.

   a. Close the **Optimization Wizard**.

   b. Add a page to the MotionView session and change the client of the newly added page to HyperGraph.

   c. Open the **abf** file in the last iteration folder of the optimization run.

   Note The optimization run directory is automatically created during the solution and follows the naming convention `py_name_timestamp` where `py_name` is the name of the python file given in the **Solution** page and `timestamp` is the time when the solution was initiated.



Look in: mv_3022_initial_4bar_opt_133030

Py_name given in Solution page appended with *_timestamp*

Name	Date modified	Type
iter-21	11/21/2018 1:31 PM	File folder
iter-22	11/21/2018 1:31 PM	File folder
iter-23	11/21/2018 1:31 PM	File folder
iter-24	11/21/2018 1:31 PM	File folder
iter-25	11/21/2018 1:31 PM	File folder
iter-26	11/21/2018 1:31 PM	File folder
iter-27	11/21/2018 1:31 PM	File folder
design.log	11/21/2018 1:31 PM	Text Docume
logfile.log	11/21/2018 1:31 PM	Text Docume
mv_3022_initial_4bar_opt.json	11/21/2018 1:31 PM	JSON File
summary.log	11/21/2018 1:31 PM	Text Docume

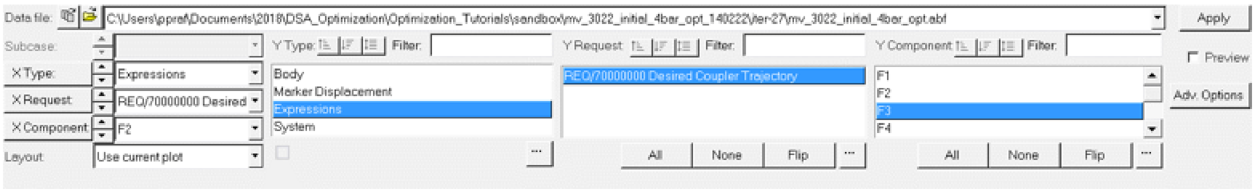
Last iteration folder has the optimized result

File name:

Files of type: All Files (*.*)

d. Select Type - Request - Component as follows:

Axis	Type	Request	Component
X	Expressions	REQ/70000000 Desired Coupler Trajectory	F2
Y	Expressions	REQ/70000000 Desired Coupler Trajectory	F3



Data file: C:\Users\ppraf\Documents\2018\DSA_Optimization\Optimization_Tutorials\sandbox\mv_3022_initial_4bar_opt_140222\iter-27\mv_3022_initial_4bar_opt.abf

X Type: Expressions

X Request: REQ/70000000 Desired

X Component: F2

Layout: Use current plot

Y Type: Expressions

Y Request: REQ/70000000 Desired Coupler Trajectory

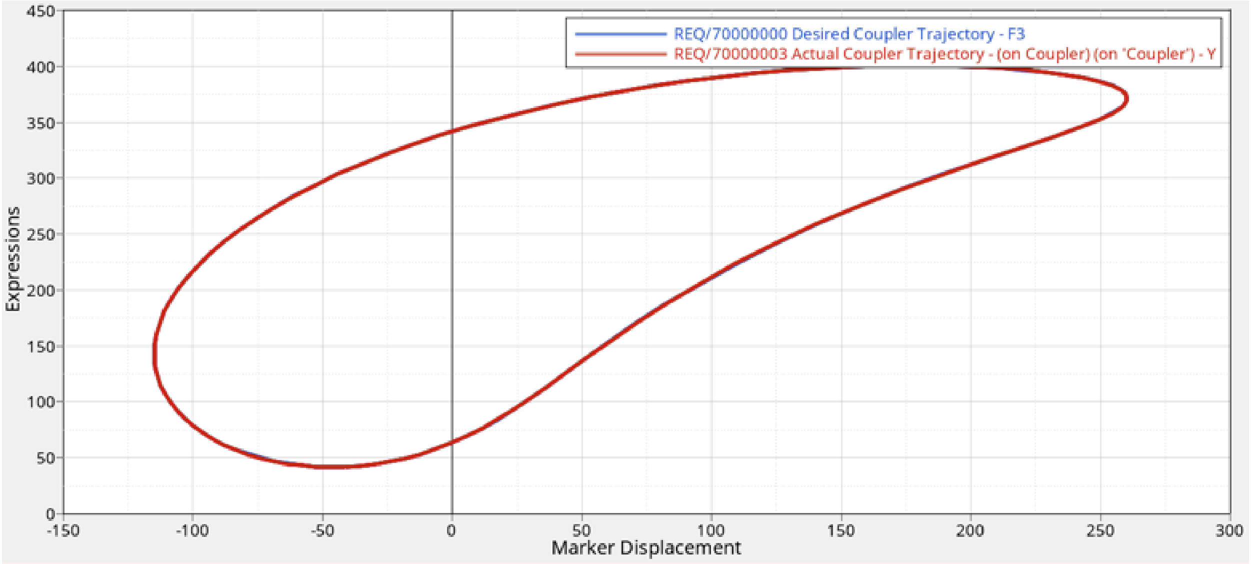
Y Component: F3

Click ***Apply***. This plots the desired profile of the trajectory for the coupler CM.

Next, plot the actual trajectory of the coupler CM by plotting the following:

Axis	Type	Request	Component
X	Marker Displacement	REQ/70000003 Actual Coupler Trajectory - (on Coupler) (on 'Coupler')	X
Y	Marker Displacement	REQ/70000003 Actual Coupler Trajectory - (on Coupler) (on 'Coupler')	Y

The resulting plots would overlay over each other.



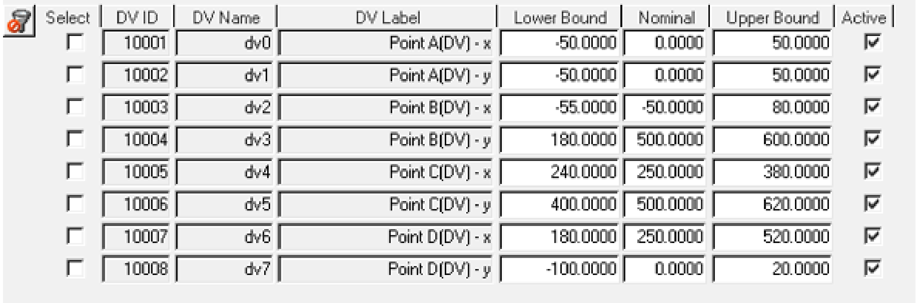
Step 7: Identifying Infeasible Designs and Possible Recovery (Optional)

In this section, we introduce a powerful feature in MotionSolve optimization that can help to get the optimal design by recovering from a failed iteration.

It is not uncommon for a mechanical system to be limited by some sort of physical constraints. In this four bar example, the Grashof constraints have to be satisfied in order to make the input link a crank. When optimizer chooses a new design, those constraints are likely to be violated regardless of whether they are added as optimization constraints or not. The optimizer in MotionSolve has a recovery mode to help recover from such infeasible design and let optimization proceed.

To demonstrate how the optimizer recovers in this tutorial, we will change the initial design and see how it works.

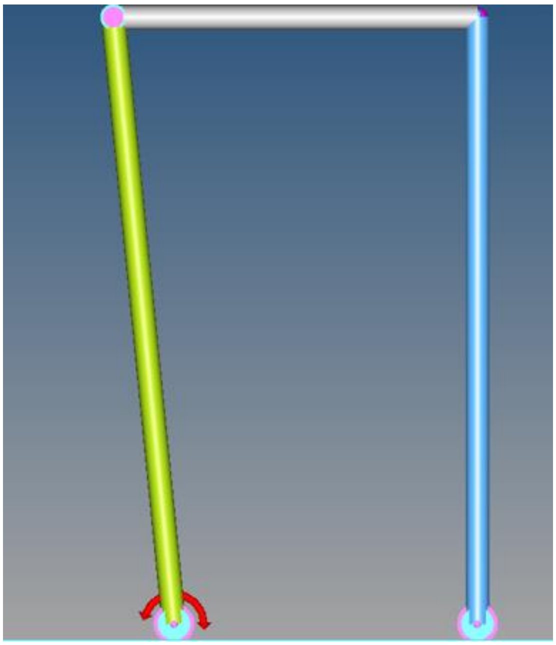
1. Go back to the **Design Variables** page and change the nominal value, upper and lower bound as follows:



Select	DV ID	DV Name	DV Label	Lower Bound	Nominal	Upper Bound	Active
☐	10001	dv0	Point A(DV) - x	-50.0000	0.0000	50.0000	☑
☐	10002	dv1	Point A(DV) - y	-50.0000	0.0000	50.0000	☑
☐	10003	dv2	Point B(DV) - x	-55.0000	-50.0000	80.0000	☑
☐	10004	dv3	Point B(DV) - y	180.0000	500.0000	600.0000	☑
☐	10005	dv4	Point C(DV) - x	240.0000	250.0000	380.0000	☑
☐	10006	dv5	Point C(DV) - y	400.0000	500.0000	620.0000	☑
☐	10007	dv6	Point D(DV) - x	180.0000	250.0000	520.0000	☑
☐	10008	dv7	Point D(DV) - y	-100.0000	0.0000	20.0000	☑

Value of Design Variables

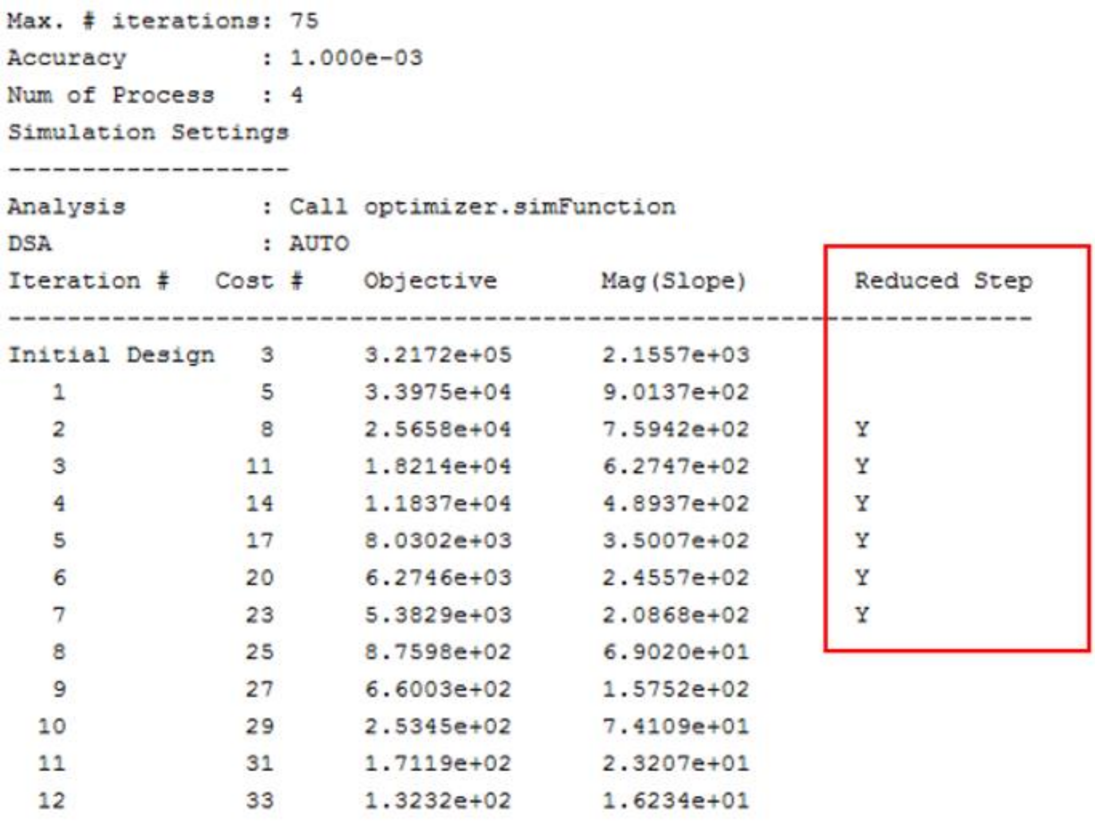
The four bar mechanism should look like below after the change:



Four bar mechanism after changing the value of Dvs

Retain the Responses, Objectives and Constraints as defined in Step 2 and 3 above.

2. Run an optimization with a different **Output file** in the **Solutions** page as `4bar_opt_for_recover.py`. The number of iterations will change due to a different initial design. Pay attention to the last column 'Reduced Step' in the solver output:

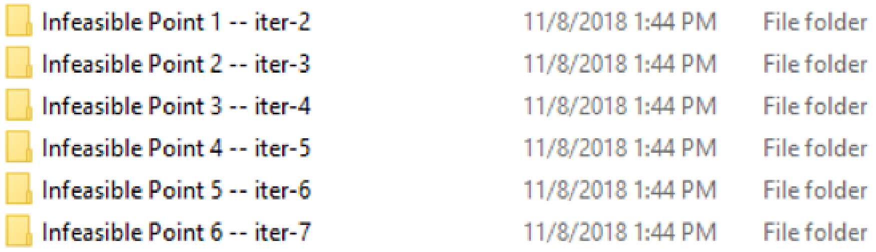


```
Max. # iterations: 75
Accuracy          : 1.000e-03
Num of Process    : 4
Simulation Settings
-------------------
Analysis          : Call optimizer.simFunction
DSA               : AUTO
Iteration #   Cost #    Objective       Mag(Slope)       Reduced Step
---------------------------------------------------------------------
Initial Design   3      3.2172e+05      2.1557e+03
   1             5      3.3975e+04      9.0137e+02
   2             8      2.5658e+04      7.5942e+02          Y
   3            11      1.8214e+04      6.2747e+02          Y
   4            14      1.1837e+04      4.8937e+02          Y
   5            17      8.0302e+03      3.5007e+02          Y
   6            20      6.2746e+03      2.4557e+02          Y
   7            23      5.3829e+03      2.0868e+02          Y
   8            25      8.7598e+02      6.9020e+01
   9            27      6.6003e+02      1.5752e+02
  10            29      2.5345e+02      7.4109e+01
  11            31      1.7119e+02      2.3207e+01
  12            33      1.3232e+02      1.6234e+01
```

Solver output when recovery mode is invoked

In this example, the optimizer has found infeasible design in iteration 2-7; it is then told to reduce the step size until a feasible design is found.

3. Check the output files. In most cases, the output files could tell you why solver fails at a specific point and how you can avoid it if possible.



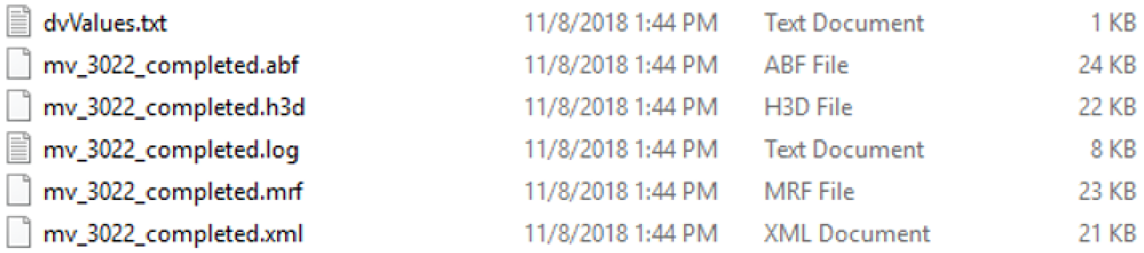
Infeasible Point 1 -- iter-2	11/8/2018 1:44 PM	File folder
Infeasible Point 2 -- iter-3	11/8/2018 1:44 PM	File folder
Infeasible Point 3 -- iter-4	11/8/2018 1:44 PM	File folder
Infeasible Point 4 -- iter-5	11/8/2018 1:44 PM	File folder
Infeasible Point 5 -- iter-6	11/8/2018 1:44 PM	File folder
Infeasible Point 6 -- iter-7	11/8/2018 1:44 PM	File folder

Subfolders that contain the infeasible designs

When the optimizer recovers from an infeasible design, an additional folder 'Infeasible Points' is created; each subfolder contains the following information:

- dvValues.txt: Value of design variables
- 4bar_opt_for_recover.abf: MotionSolve output file
- 4bar_opt_for_recover.h3d: Animation file (provided only when available)
- 4bar_opt_for_recover.log: MotionSolve log file

- 4bar_opt_for_recover.mrf: MotionSolve output file
- 4bar_opt_for_recover.xml: MotionSolve xml input file

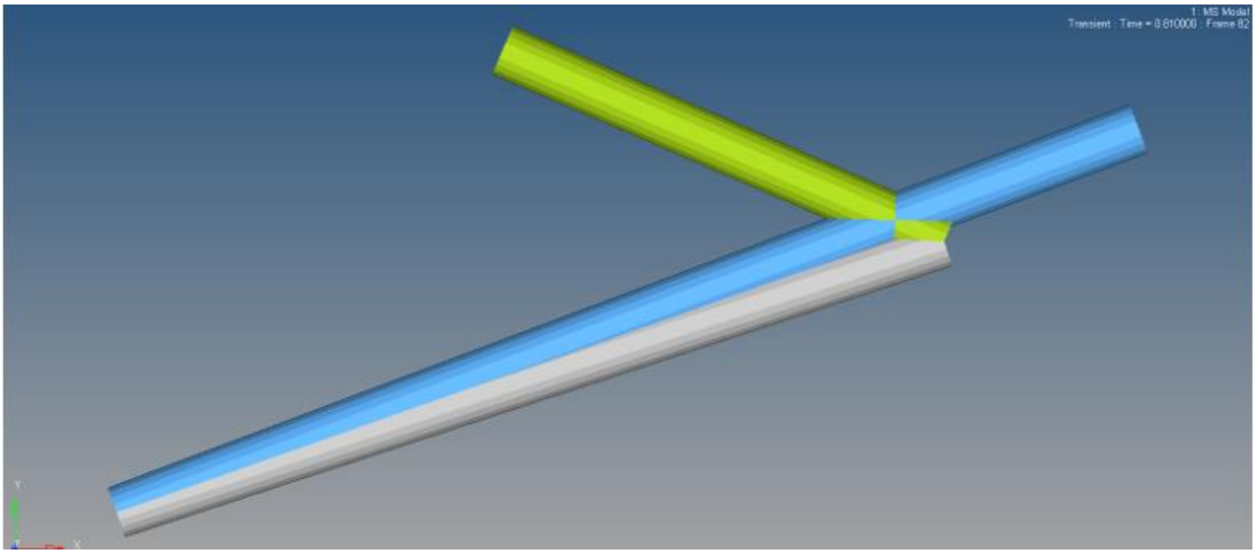


dvValues.txt	11/8/2018 1:44 PM	Text Document	1 KB
mv_3022_completed.abf	11/8/2018 1:44 PM	ABF File	24 KB
mv_3022_completed.h3d	11/8/2018 1:44 PM	H3D File	22 KB
mv_3022_completed.log	11/8/2018 1:44 PM	Text Document	8 KB
mv_3022_completed.mrf	11/8/2018 1:44 PM	MRF File	23 KB
mv_3022_completed.xml	11/8/2018 1:44 PM	XML Document	21 KB

Output for each infeasible design

We have found that the h3d file and log file are useful in most cases. We will load the h3d file in this example in HyperView and see what is going wrong.

4. Load `4bar_opt_for_recover.h3d` in folder **Infeasible Point 1 -- iter-2**. You can see that the system enters a lock-up position before the solver crashes:



Last frame before solver crashes

5. Check the result. The result we get here is very close to what we get in STEP 5 and the cost function is reduced significantly. This shows that we are able to get the optimal design even with infeasible design encountered in the intermediate step. However, the optimality is not guaranteed in some cases. When that happens, you can improve the result by adding proper constraints based on your investigation in previous steps. For four bar mechanism, Grashof constraints can be added and we provide an example that shows you how to do it in MotionView.