# Motivation

In short, we want to calculate a “sacrifice ratio” to be used as a proxy for model stickiness, so that we can include it in our regressions. The sacrifice ratio looks at how much output must be sacrificed to permanently reduce inflation. As Bob put it:

Bob: “I’m trying to think of ways to come up with new RHS variables for our regressions that will allow a clear message to show through. One of the things that occurred to me was that we could compute a sacrifice ratio for each model and use that. The logic here is that our experiments to date are ostensibly about the “monetary policy multiplier,” which is about the power of MP in a particular context, one where the authority introduces a shock and then undoes what it started. Computing a sacrifice ratio is a different kettle of fish. The sacrifice ratio is the (cumulative, annualized) output cost of disinflation, computed from an experiment where the monetary authority reduces the target rate of inflation, once and for all. For me, at least, the key thing about this experiment is that the CB does not clean up the mess it causes.

To do these experiments, in principle, we need to adjust the policy rule that we’re running to include a target rate of inflation, set the target lower by one percentage point permanently at the outset of the experiment, run a simulation and then compute and record the sacrifice ratio as follows: cumsum(outputgap[t]-outputgap\_baseline[t])/4)/(inflation[t=T]-inflation[t=0]) where the experiment is run from t=0 to T.”

# Methodology

## Overall

To do this, we need to implement a target inflation (pitarg) variable into the model equations for every single model, shock it permanently, record the IRFs, and then calculate the sacrifice ratios.

## SED to Adjust .mod Files

Since pitarg is not in any of the models in MMD by default, we must add it. Specifically, we must add an endogenous variable called “pitarg” (along with its equation), an exogenous shock to pitarg called “shk”, and put pitarg in the model equation.

Note, we have to write the pitarg equation such that it equals itself in the previous period plus a shock. Thus, if we shock it downwards once, it will be permanently lower by the size of that shock. This is indistinguishable from just having a negative pitarg shock itself.

But how can we make this change to every file without manually going in to each one? We use the [SED](https://www.gnu.org/software/sed/manual/sed.html) utility in Linux. Specifically, we run:

* sed -e '/Modelbase Variables \*\\//a pitarg' -e '0,/varexo/{s/varexo/varexo shk/}' -e \"s|Modelbase Variables in Terms of Original Model Variables|Modelbase Variables in Terms of Original Model Variables\npitarg = pitarg(-1) - shk;|\" -e '/shocks;/a var shk;\nstderr 0.25;' -e 's/\*interest\_/\*interest\_ -(cofintinf0+cofintinfb1+cofintinfb2+cofintinfb3+cofintinfb4+cofintinff1+cofintinff2+cofintinff3+cofintinff4)\*pitarg/' -e '$astoch\_simul(irf=60, nograph);' -e '$asave(\"[model]\_pitarg.mat\")' [model].mod > ../pitarg\_models\_raw/[model]\_pitarg.mod"

That looks crazy. What is actually happening? Let’s break it down

* You can string multiple sed commands together with the “-e” option, so I can do sed -e step 1 -e step2 -e step3. This is why there are otherwise random “-e” appearances in there
* '/Modelbase Variables \*\\//a pitarg'
  + This says find where there are the words “Modelbase Variables”, any number of spaces (represented in regex by “ \*”), and then two forward slashes (represented in regex by “\\//”). Where that pattern exists, append the word pitarg.
  + This is to initialize pitarg as an endogenous variable in the models. I had previous relied on the word “var”, but that would sometimes occur in comments before it appeared in the actual code, so pitarg was only being inserted into comments
  + I also tried to just append after “Modelbase Variables”, but that caused issues because down further in every file is the sentence “Definition of Modelbase Variables in Terms of Original Model Variables”
* '0,/varexo/{s/varexo/varexo shk/}'
  + This says: look in the range from beginning (0) of the file to the first instance of “varexo”, and in that range replace “varexo” with “varexo shk”
    - This adds the exogenous shk shock
* -e \"s|Modelbase Variables in Terms of Original Model Variables|Modelbase Variables in Terms of Original Model Variables\\npitarg = pitarg(-1) + shk;|\"
  + This says: When you come across the phrase “Modelbase Variables in Terms of Original Model Variables” you should add a new line (“\n”) and then the pitarg equation (pitarg = pitarg(-1) + shk)
  + I use this phrase because it is the only phrase common to all files that lets me put something in the model section
* '/shocks;/a var shk;\nstderr 0.25;'
  + Where you see “shocks” you should append (‘a’) “var shk (new line, using \n) stderr 0.25”
  + Note: When running directly in the shell, this syntax is fine. However, when running in R, I need to put \\n. Why double backslash? Because otherwise R itself sees the \n and tries to execute a return. The double back slash says hey R, just send the \n to the shell.
* 's/\*interest\_/\*interest\_ -(cofintinf0+cofintinfb1+cofintinfb2+cofintinfb3+cofintinfb4+cofintinff1+cofintinff2+cofintinff3+cofintinff4)\*pitarg/'
  + This says: find the pattern “\*interest\_” and replace it with \*interest\_ and then all the coefficients times pitarg, which is how Bob said to implement pitarg into the policy rule.
* '$astoch\_simul(irf=60, nograph);'
  + This says go to the end of the document ($) and append (a) stoch\_simul(irf=60, nograph);
* '$asave(\"[model]\_pitarg.mat\")'
  + This says go to the end of the document ($) and append (a) save(\"[model]\_pitarg.mat\")
  + [model] tells glue to go to whatever model we are currently on and use that name. \ is just an escape for the quotation marks
  + Note: This only works in R. The brackets are used by glue() to realize what is inside them is a variable. So “[model]” is replaced with something like CGG99\_test
* [model].mod > ../pitarg\_models\_raw/[model]\_pitarg.mod"
  + This says: run all the commands on model.mod (where [model] is the given model being adjusted at the given time) and save it in the pitarg\_models\_raw/ directory under the name model\_pitary.mod (again, where [model] is the given model being altered)

How do I actually implement this? R allows one to interact with the shell. So I created an R script in which I run a loop across all models, for each one running this SED command on their .mod file. The script is called: /msu/res1/m1jps07/Bob/IRF\_Project/Calc-Sacrifice-Ratio/01\_Add-Pitarg.r. After running this script, the files are stored in Calc-Sacrifice-Ratio/pirtarg\_models/. Note, in case someone accidentally overwrites them, the last ones I used when running the regressions are stored within that directory in the backup/ folder.

**Note:** While the above SED code does make the proper adjustments to most files, it is possible there will be some outliers. Unfortunately, there is no proper documentation of those outliers at this time. Thus, if one of the modified .mod files that we want to run to calculate the sacrifice ratio does not work, check to make sure SED did what we expected/wanted it to. Sometimes a given .mod file will have a unique structure that makes my regular expressions not work.

## Run .mod Files

* This will generate the IRFs that we use to calculate the sacrifice ratio
* Run “Run\_Dynare.m”. A few notes:
  + The Run\_Dynare.m file itself is in Calc-Sacrifice-Ratio/. However, when you open it, you’ll want to set the working directory in Matlab to the pitarg\_models/ subdirectory. This way Matlab knows where to look for the .mod files
  + You will need to initialize Dynare for that session by running “addpath c:\dynare\4.5.4\matlab”
    - See [here](https://archives.dynare.org/documentation-and-support/quick-start.html) for details
* This will run each model and save a series of outputs in pitarg\_models/
  + We only need the .mat files, and we want to save them in a folder than indicates what rule was used. For example, if the Taylor rule had been used, we want to save the .mat files in results-taylor-rule/
    - For the other output files, they are mostly useless. I would navigate to the Calc-Sacrifice-Ratio/ folder, using the shell in linux, and run “mv pitarg\_models/\*.m Run-Outputs/” and “mv pitarg\_models/\*.log Run-Outputs/” to move them to a placeholder folder called “Run-Outputs/”

## Extract/Calculate Sacrifice Ratios

* Navigate to run 02\_Extract-Sacrifice-Ratios.r
* Select the rule you used when running the .mod files
  + The main rule being used as of this writing is the “bob” rule
* Simply run the script. Follow the detailed comments for explanations of each step

## Errors

* Out of the 90 models of interest, we were able to obtain sacrifice ratios for 79 of them (87.8%).
  + Of the 11 we don’t have sacrifice ratios for:
    - 1 model has the “impossible to find steady state” error
      * NK\_DEFK17\_pitarg
    - 8 models have the “NA or NaN in steady state” error
      * NK\_DT12\_pitarg
      * NK\_ET14\_pitarg
        + The non-modified .mod file does *not* run, the modified one does not
      * NK\_FLMF18\_pitarg
        + The non-modified .mod file does run, the modified one does not
      * NK\_GK11\_pitarg
      * NK\_GSSZ17\_pitarg
      * NK\_MM10\_pitarg
      * US\_FV15\_pitarg
      * US\_IR15\_pitarg
    - 1 models has the “Blanchard Kahn conditions are not satisfied: indeterminacy” error
      * US\_CCF12\_pitarg
    - 1 model has an “Incorrect dimensions for matrix multiplication” error
      * G7\_TAY93\_pitarg
        + Note: .mod file here loads Sigma\_G7\_TAY93.mat
        + The error is: “Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To perform elementwise multiplication, use '.\*'.”
        + This error occurs only in the pitarg modified version of the file, not in the basic version. So something about adding pitarg breaks the model, but it is not clear the juice is worth the squeeze, at this moment, in terms of figuring this out

# Q and A

## What do the different folders hold?

* data/ holds the sacrifice ratio data
* figures/ holds the figures
* helper-mat-files/
  + This folder holds the .mat files used by some of the .mod files. In addition to the policy-params.mat file, some .mod files need to call other variables, which are held in these .mat files
* models/
  + This holds the *raw*, unmodified .mod files for the models. The .mod files used by MMB
* models-of-interest\_in\_folders/
  + This holds not only raw, unmodified .mod files from MMB, but the folders they came in, including any supporting .mat files and the .json files
* old.misc/
  + This holds outdated or temporary files
* pitarg\_models/
  + This holds the modified .mod files (with pitarg implemented) and other misc adjustments made manually
  + This also holds the .mat files for each of the .mod files under different rules, held within subfolders called results-rulename
  + It also holds a backup folder with the .mod files with pitarg implemented, just in case one accidently overwrites the others
* pitarg\_models\_raw/
  + This holds the .mod files exactly as they were after we ran the SED utility on them, with no manual modifications
* policy-params-files/
  + This holds the .mat files with the coefficients for different rules. For example, the .mat file with the coefficients for the Taylor rule is held here
* Run-Outputs/
  + This holds the extraneous files created when we run the .mod files. Just a place to keep them without deleting
* testing-station/
  + This is where one can run tests using SED, running .mod files, and so on

## What rule is used? What rules could be used?

* The rule used as of the timing of this writing (12/1/2021) is the “Bob” rule.
  + This has a coefficient of 0.0938 on the t through t-3 values of inflation
  + Coefficient of 0.5 on the t value of output
* The rest of the rules that can be used can be found in Calc-Sacrific-Ratio/policy-params-files/
  + This is where we save the .mat files containing the coefficients

## What if I want to change the rule being used?

* Go into Calc-Sacrifice-Ratio/policy-params-files/ and find the .mat file for the rule you want (Bob rule, Taylor rule, etc.)
* Copy this .mat file and paste it in Calc-Sacrifice-Ratio
* Change the name to policy-param.mat
* Then you can run the .mod files and they will look to this new .mat file containing the coefficients for the rule of interest
* If you want to do a rule not previously done, copy one of the policy-param.mat files and just adjust the coefficients as desired. Then, paste this new file in Calc-Sacrifice-Ratio/ with the name policy-param.mat

## What is the policy\_param.mat file?

* This file is basically a saved Matlab workspace that sets up the rule we use (Taylor, Inertial, etc.)
* *All* we want from it, when running a given job, are the coefficients (based on the rule we are using) and std\_r\_ and std\_r\_quart
  + I say this because sometimes this file may be overwritten and contain all the workspace variables from a previously run job. This can lead to serious issues. For example:
    - If there is a “thispath” variable from a previous run saved, the currently running job will try to use that pathway rather than the pathway of the current job itself. Thus, if the path from which you run the job is “Jake”, but the “thispath” variable saved in policy\_param.mat is “Scott”, it will try to go to the “Scott” folder, leaving confusion for everyone involved. Thus, we should always clean up policy\_param.mat before a run
* When you run a job, by running one of the model’s .mod files, it looks at the current directory, goes up one step, and then grabs the policy\_param.mat values. Thus, you should always ensure that policy\_param.mat is one level above the .mod files you’re running

## What is the helper-mat-files folder?

This folder holds the .mat files used by some of the .mod files. In addition to the policy-params.mat file, some .mod files need to call other variables, which are held in these .mat files

## What if we see “error using load” in Matlab when trying to run a model?

This means the model you are running needs to load a .mat file besides policy-param.mat in order to run. These .mat files are stored in helper-mat-files/ and should be copy and pasted into pitarg\_models/ when running such models in order to avoid such errors.

For example, US\_HL16 requires median\_param\_BEP.mat. Normally, this is stored only in helper-mat-files/. But when I want to run US\_HL16, I will copy median\_param\_BEP.mat into pitarg\_models/. Then, when done, I delete median\_param\_BEP.mat.

## How do I install/use Dynare? Matlab?