MSE\_Mathematica

MSE in Mathematica v.1.0.0

# Contents

[**Contents**](#_7ph9p3rkpfzz)[**1**](#_7ph9p3rkpfzz)

[**Introduction**](#_b7lx46tymnzg)[**3**](#_b7lx46tymnzg)

[Improvements & technical aspects](#_jkg770t4hhar) [3](#_jkg770t4hhar)

[**Data files structure and importing functions**](#_xscaaqv8y9so)[**5**](#_xscaaqv8y9so)

[Stream (“stream”)](#_i16psvu5az9b) [6](#_i16psvu5az9b)

[Precomputed (“precomp”)](#_daac8rg7hhx2) [6](#_daac8rg7hhx2)

[**Variables and their data structure**](#_6vv3hm1lc3a6)[**6**](#_6vv3hm1lc3a6)

[Input Variables](#_fmch5cilbjch) [6](#_fmch5cilbjch)

[Global variables that their names should not be changed.](#_4ifnl0gsnofw) [7](#_4ifnl0gsnofw)

[Data structures](#_oeumm1w5g2i3) [7](#_oeumm1w5g2i3)

[Settings](#_ovyijwstdve) [8](#_ovyijwstdve)

[**Use case Flows**](#_5mgg2ao6wx7t)[**8**](#_5mgg2ao6wx7t)

[Flow 1: Find optimum match for all Markets given a specific payoff function](#_lfir47i7mrl7) [8](#_lfir47i7mrl7)

[Description](#_hwz5swhzn25g) [8](#_hwz5swhzn25g)

[Diagram](#_sqnfudll07yv) [8](#_sqnfudll07yv)

[Examples](#_eak0d7ss8nh3) [8](#_eak0d7ss8nh3)

[Flow 2: Maximize the parametric payoff function based on the observed matches](#_mpu5z4lh239q) [9](#_mpu5z4lh239q)

[Description](#_85y54xdgxrk4) [9](#_85y54xdgxrk4)

[Diagram](#_67jubirh5762) [9](#_67jubirh5762)

[Examples](#_phik2petqk64) [9](#_phik2petqk64)

[Flow 3: Calculate the stability of the calculated payoff function](#_qtryn2gbgzmf) [9](#_qtryn2gbgzmf)

[Description](#_pbypxcnsuvsn) [10](#_pbypxcnsuvsn)

[Example](#_8mwya5mfdux9) [10](#_8mwya5mfdux9)

[More Flows… The freedom to create your own flow.](#_mfs1pbo8l09v) [10](#_mfs1pbo8l09v)

[**Appendix 1: Functions Reference**](#_kj83gqgjaq2w)[**10**](#_kj83gqgjaq2w)

[File: import.m](#_x9p6f0x51kmm) [11](#_x9p6f0x51kmm)

[import](#_cp6ma8rm3pf2) [11](#_cp6ma8rm3pf2)

[File: payoff.m](#_gpmtcz4rar38) [11](#_gpmtcz4rar38)

[Cx](#_fepx2hmkizbz) [11](#_fepx2hmkizbz)

[payoff](#_sgoel7gkpie2) [12](#_sgoel7gkpie2)

[payoffDM](#_d1phr68acut5) [12](#_d1phr68acut5)

[CpayoffMatrix](#_d8vb92qtnfus) [13](#_d8vb92qtnfus)

[Ctotalpayoff](#_osky0e79q71v) [13](#_osky0e79q71v)

[File: matching.m](#_cvc0dkixvle0) [14](#_cvc0dkixvle0)

[generateAssignmentMatrix](#_i82z3ctshmre) [14](#_i82z3ctshmre)

[CmatchMatrix](#_ms1hxzth03pr) [14](#_ms1hxzth03pr)

[Cmates](#_ei0a11staaxf) [15](#_ei0a11staaxf)

[Cmate](#_v3qqamqso7y0) [15](#_v3qqamqso7y0)

[quotas](#_qqcvat7916cu) [16](#_qqcvat7916cu)

[File: inequalities.m](#_sh3sqs7fq266) [16](#_sh3sqs7fq266)

[Cineqmembers](#_wy6stkxdmy92) [16](#_wy6stkxdmy92)

[Cinequalities](#_wqnwe8rykkj0) [17](#_wqnwe8rykkj0)

[File: dataArray.m](#_7ozwa1x3snxc) [17](#_7ozwa1x3snxc)

[CdataArray](#_7nbh7zsc22z0) [17](#_7nbh7zsc22z0)

[File: objective.m](#_unuthl1xspbg) [18](#_unuthl1xspbg)

[coefficient1](#_68i3pkpr4lh6) [19](#_68i3pkpr4lh6)

[objective](#_n0dh4rl5nz5f) [19](#_n0dh4rl5nz5f)

[File: maximize.m](#_te7kgdjcsdp5) [20](#_te7kgdjcsdp5)

[optimize](#_bb2js28jduky) [20](#_bb2js28jduky)

[maximize](#_d9mlx4e0zeff) [21](#_d9mlx4e0zeff)

[File: confidence.m](#_mspdsg6bb9nk) [22](#_mspdsg6bb9nk)

[generateRandomSubsample](#_qc7ho5v9n1mx) [22](#_qc7ho5v9n1mx)

[pointIdentifiedCR](#_gqoxq5y93e8s) [22](#_gqoxq5y93e8s)

[**Appendix 2: Code style in Mathematica**](#_mksx9xmz67u)[**23**](#_mksx9xmz67u)

[**Appendix 3: Testing all functions**](#_ouhtle98ao9g)[**24**](#_ouhtle98ao9g)

[**Appendix 4: Optimization Methods**](#_xkzgbsbd3m4m)[**24**](#_xkzgbsbd3m4m)

[**Appendix 5: Implementation in more Programming languages**](#_hqdftx2892nv)[**25**](#_hqdftx2892nv)

# 

# 

# Use case Flows

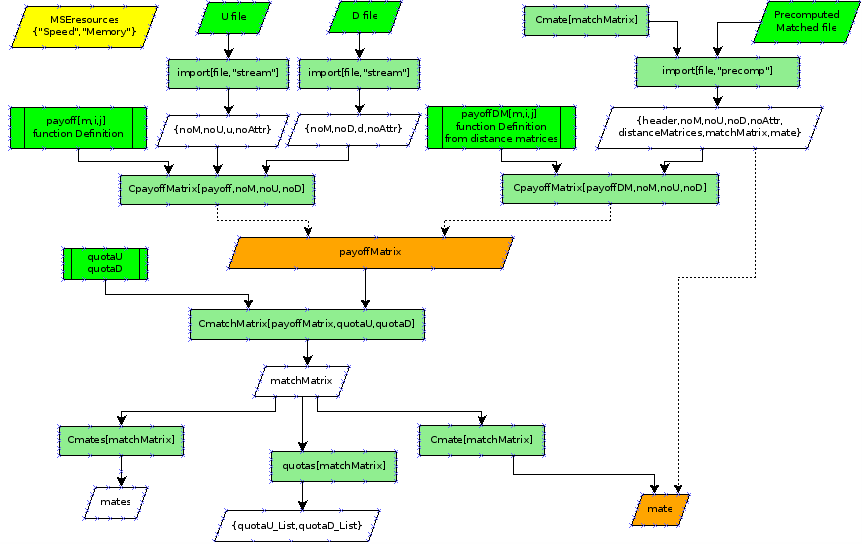
## Flow 1: Find optimum match for all Markets given a specific payoff function

### Description

In this flow we suppose that we have imported files that contain data (stream or precomputed). In the case of precomputed we already know the payoffMatrix.

We set the quotas per Market / Stream / Specific member that puts the constrains on how many matches each individual can create.

### Diagram



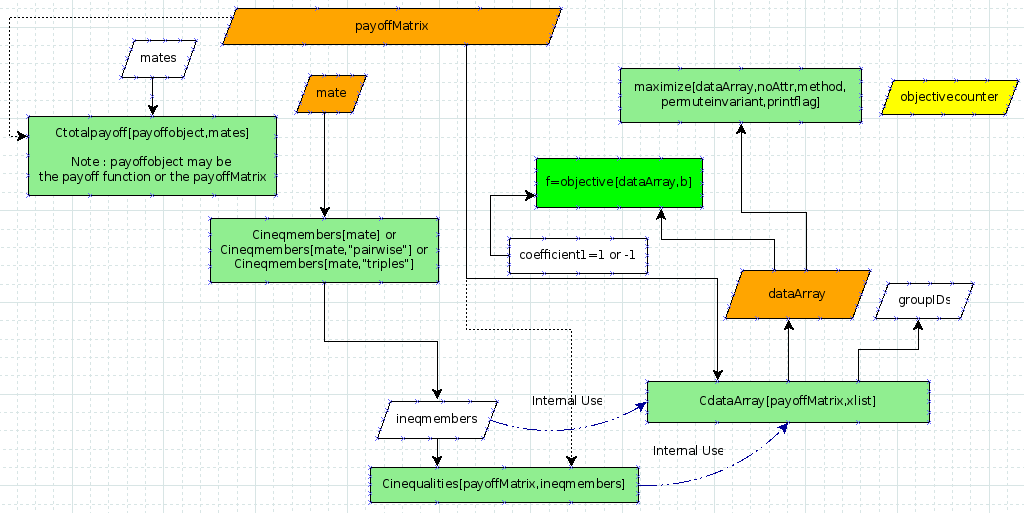
### Examples

## Flow 2: Maximize the parametric payoff function based on the observed matches

### Description

In this flow we suppose that the dataArray is already created and of cource the payolff function is known. The aim is to find the parameters of the payoff function (the b-list) in order to maximize the number of satisfied inequalities.

### Diagram



### Examples

docs/precomputed\_abstract.nb (ends on forming the dataArray - abstract values)

docs/precomputed\_numeric.nb (full flow)

## Flow 3: Calculate the stability of the calculated payoff function

### Description

In this flow we suppose that we have calculated the parameters of the payoff function that maximizes the number of satisfied inequalities. We aim to test the stability of this solution through calculating the confidence intervals.

The process we are following is the known one, ommiting markets - find the new parameters that maximize the new dataArray and do it several times in the way it is describe in the code or in Jeremy’s doc.

### Example

## More Flows… The freedom to create your own flow.

Since the code is modular, that is it consist of autonomous functions one can combine them in a free way that makes sence.

So someone can define new functions that will be able to munipulate the data at any point of the flow in order to study for various patterns.

An example would be to remove strong pairs from the upstream and downstream sets and study how the Market would evolve without them. This is interesting especially in the many to many relationships.

If you are interested in building your own flow, please send an e-mail to …. to guide you through.