# Usermanual

Version: December 9, 2022

# IPbasic

Author:
Andreas Huber*

info@ocpbasic.com, www: http://www.ocpbasic.com

# Contents

# 1 Licence

## 1.1 Commercial Licence

To get more information about the commercial licence and the commercial use of IPBASIC and OCPBASIC please contact info@ocpbasic.com.

## 1.2 Free Open Source Licence LGPL v3

### GNU LESSER GENERAL PUBLIC LICENSE



This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

   As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

   "The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

   An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

   A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

   The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

   You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

   If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

   a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

   The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

   You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

# 2 Installing IPbasic and OCPbasic

## 2.1 Fast Installing Procedure

If you do not like to build IPBASIC and OCPBASIC yourself you can get precompiled packages for Ubuntu 22.04.

### 2.1.1 Installing via apt-get on Ubuntu 22.04

This section will install IPBASIC, OCPBASIC and LAPACK_WRAPPER using an apt-repository and the linear algebra packages libblas-dev and liblapack-dev.
First download and activate the ocpbasic gpg key and install it to your system.

**Comment 2.1** *Only add software repositories from sources that you trust! Third-party software repositories are not checked for security or reliability and may contain software which is harmful to your computer. The use of these repositorys is at your own risk!*

```
$ wget https://archive.ocpbasic.com/dists/jammy/ocpbasic.gpg
$ sudo install -o root -g root -m 644 ocpbasic.gpg \
 /etc/apt/trusted.gpg.d/
$ sudo rm ocpbasic.gpg
```

After that you can activate the ocpbasic apt-repository with the command:

```
$ sudo add-apt-repository \
'deb␣[arch=amd64]␣http://archive.ocpbasic.com/dists/jammy␣/'
```

If the apt-repository is installed correctly you can install OCPBASIC with the command:

```
$ sudo apt-get install ocpbasic
```

The software packages libblas-dev, liblapack-dev, LAPACK_WRAPPER and IPBASIC will then also be installed by apt.
**Supported Ubuntu Versions:**

| **Ubuntu version:** | **apt-repository** |
|---|---|
| Ubuntu 22.04 | 'deb [arch=amd64] http://archive.ocpbasic.com/dists/jammy /' |
| Ubuntu 20.04 | 'deb [arch=amd64] http://archive.ocpbasic.com/dists/focal /' |

| | |
|---|---|
| Ubuntu 18.04 | 'deb [arch=amd64] http://archive.ocpbasic.com/dists/bionic /' |

Table 1: Supported Ubuntu Versions

**Comment 2.2** *The maintainer may remove or delete these apt-repositorys any time!*

## 2.2 Selection of linear algebra software packages

| Package: | BLAS | LAPACK |
|---|---|---|
| **BLAS / LAPACK:** | ✓ | ✓ |
| **OpenBLAS** | ✓ | ✓ |
| **Intel Math Kernel Library (MKL)** | ✓ | ✓ |
| **ATLAS** | ∼ | ∼ |
| **Accelerate - Apple Developer** | ∼ | ∼ |

Table 2: Dense linear algebra software packages used by LAPACK_WRAPPER

Table 2 shows all tested and supported dense linear algebra software packages from LAPACK_WRAPPER, that can be used by IPBASIC and OCPBASIC.

- ✓ Supported by LAPACK_WRAPPER and tested with IPBASIC and OCPBASIC.
- ∼ Supported by LAPACK_WRAPPER but not tested with IPBASIC and OCPBASIC.

### 2.2.1 Selection of linear algebra software packages using qmake

**Comment 2.3** *The user is responsible for linking against any thirdparty software and checking the licence compatibility.*

When using the default build method qmake for building LAPACK_WRAPPER, IPBASIC and OCPBASIC you will find a config.pri file in the src/ folder. In this file you will find a section SET DEFINES, where used packages can be selected. Fild the following lines:

```
#
#  Define used linear Algebra.
#
# Only supported by lapack_wrapper:
```

```
#  LAPACK_WRAPPER_USE_ACCELERATE, LAPACK_WRAPPER_USE_ATLAS,
#
# Currently supported values by IPbasic and OCPbasic:
#  LAPACK_WRAPPER_USE_LAPACK, LAPACK_WRAPPER_USE_OPENBLAS,
#  LAPACK_WRAPPER_USE_MKL
#
DEFINES += LAPACK_WRAPPER_USE_LAPACK
```

You can change LAPACK_WRAPPER_USE_LAPACK into your prefered linear algebra software package define.

**Comment 2.4** *You must only use one single linear algebra software package define!*

After selecting your linear algebra software package you have to set the include and shared library path and the linked linear algebra package. Terefore you will find a section SET PATH in the config.pri file.

```
#
# Define lib and include path for blas and lapack:
#
BLAS_INCLUDE_PATH = /usr/include/x86_64-linux-gnu/

BLAS_PATH_LIBS = -L/usr/lib/x86_64-linux-gnu/ -lblas

LAPACK_INCLUDE_PATH = /usr/include/x86_64-linux-gnu/

LAPACK_PATH_LIBS = -L/usr/lib/x86_64-linux-gnu/ -llapack

#
# Define lib and include path for Openblas:
#
OPENBLAS_INCLUDE_PATH = \
        /usr/include/x86_64-linux-gnu/openblas-pthread/

OPENBLAS_PATH_LIBS = -L/usr/lib/x86_64-linux-gnu/ \
                     -lopenblas

#
# Define lib and include path for intel mkl:
#
```

```
MKL_INCLUDE_PATH = /usr/include/mkl/

MKL_PATH_LIBS = -L/usr/lib/x86_64-linux-gnu/ \
                -lmkl_intel_lp64 \
                -lmkl_intel_thread \
                -lmkl_core \
                -lmkl_rt \
                -liomp5
```

**Comment 2.5** *Do not delete or comment any unused include and shared library path. When not selected as define the path variables for other linear algebra software packages are ignored anyway.*

If you need to link additional software packages and libraries you can activate the define OCPBASIC_ADD_LINKED_LIBS by uncommenting the following line in the section SET DEFINES.

```
#
# Define use of additional linked libs.
#
DEFINES += OCPBASIC_ADD_LINKED_LIBS
```

After activating the define OCPBASIC_ADD_LINKED_LIBS you can link additional libraries in the variable ADD_LINKED_LIBS in section SET PATH of the config.pri file.

```
#
# Additional linked libs:
#
ADD_LINKED_LIBS = -lgfortran
```

### 2.2.2 Selection of linear algebra software packages using other buildsystem

**Comment 2.6** *The user is responsible for linking against any thirdparty software and checking the licence compatibility.*

The linear algebra software packages can be selected in the configuration file of LAPACK_WRAPPER lapack_wrapper_config.hh. Uncomment the linear algebra software package define.

```
/*\
! values
!   LAPACK_WRAPPER_USE_ACCELERATE, LAPACK_WRAPPER_USE_ATLAS,
```

```
!   LAPACK_WRAPPER_USE_OPENBLAS, LAPACK_WRAPPER_USE_LAPACK,
!   LAPACK_WRAPPER_USE_MKL
\*/
#define LAPACK_WRAPPER_USE_LAPACK 1
```

Make sure you have your the corresponding linear algebra software package installed and linked with your buildsystem.

## 2.3 Selection supported of linear algebra solvers

**Comment 2.7** *By default a* LAPACK *banded KKT solver can be used with* OCPBASIC*. When only using* OCPBASIC *with its* LAPACK *banded KKT solver you do not need to install any other linear solver, which is not included in* LAPACK*.*

| **Package:** | IPBASIC | OCPBASIC |
|---|---|---|
| **HSL** | ✓ | ✓ |
| **LAPACK banded KKT** | **x** | ✓ |
| **Linear solver interface** | ✓ | **x** |

Table 3: Linear algebra solver packages used by IPBASIC and OCPBASIC

IPBASIC provides a flexible linear solver interface, where users can define own linear solvers, which are then used by IPBASIC. OCPBASIC provides a powerfull structure exploiting linear solver for KKT systems using LAPACK.
For speedtesting also an interface for the sparse linear solvers of HSL is included in IPBASIC. If activated in IPBASIC, MA57 from HSL can also be used in OCPBASIC. Since IPBASIC and OCPBASIC do not provide any copy of HSL, by default the support of HSL and MA57 is not active. If you do have a copy of HSL and you are using qmake, you can activate the support in the SET DEFINES section of the file config.pri by uncommenting the IPBASIC_USE_HSL define.

**Comment 2.8** *The user is responsible for linking against any thirdparty software and checking the licence compatibility.*

```
#
# Define use of hsl MA57 and MA48. Make sure a libhsl.so
# with all methods of MA57 and MA48 is installed to your
# system.
#
```

```
#DEFINES += IPBASIC_USE_HSL
```

Do not forget to also set a path to the hsl.so library. This can by done in the SET PATH section of the file config.pri.

```
#
# Define lib path for hsl:
#
HSL_LIB_PATH = /usr/lib/
```

If you do have a copy of HSL and you are using an other buildsystem, you can activate the support in the file IPbasic_Config.h.

```
// Flag for use of hsl:
#define IPBASIC_USE_HSL
```

## 2.4 Building and Installing IPbasic and OCPbasic on Ubuntu 20.04

### 2.4.1 Prepare buildsystem on Ubuntu 20.04

As default qmake is used for building IPbasic and OCPbasic. The installing procedure was tested on a fresh Ubuntu 20.04 LTS installation. The buildsystem can by installed by using the shell commands:

```
$ sudo apt-get install g++ -y
$ sudo apt-get install build-essential -y
$ sudo apt-get install qt5-default -y
```

**Comment 2.9** *The user is responsible for installing and linking against any thirdparty software and checking the licence compatibility.*

### 2.4.2 Install preferred linear algebra package

#### Lapack and Blas

Installing lapack and blas by using the shell command:

```
$ sudo apt-get install libblas-dev liblapack-dev -y
```

#### OpenBLAS

Installing OpenBLAS by using the shell command:

```
$ sudo apt-get install libopenblas-dev -y
```

#### Intel MKL

Installing Intel MKL by using the shell command:

```
$ sudo apt-get install intel-mkl-full -y
```

**Comment 2.10** *The user is responsible for installing and linking against any thirdparty software and checking the licence compatibility.*

**Comment 2.11** *After installing the linear algebra packages you have to change to the correct linear algebra package define and enter the correct path variables in the config.pri file as described in section 2.2.1.*

### 2.4.3 Build and install lapack_wrapper, IPbasic and OCPbasic

Open a terminal in the OCPbasicMaster/ folder. First create a build folder and change your directory into the build folder.

```
$ mkdir build
$ cd build
```

Use qmake to generate a Makefile from the OCPbasicMaster/ folder.

```
$ qmake ../
```

Build LAPACK_WRAPPER, IPBASIC and OCPBASIC using make.

```
$ make
```

Install LAPACK_WRAPPER, IPBASIC and OCPBASIC system wide using make.

```
$ sudo make install
```

After installing LAPACK_WRAPPER, IPBASIC and OCPBASIC you can check the installation using make.

```
$ make check
```

**Comment 2.12** *The command make check will execute some unittests. Some unittests return errors, if* HSL *is not used. Anyway by default* OCPBASIC *is compiled without* HSL *and you can use the* LAPACK *banded KKT solver instead.*

# 3 Using IPbasic

IPBASIC is an interior-point optimizer for nonlinear optimazation problems. It is using a sparse matrix layout defined by the software package LAPACK_WRAPPER. IPBASIC is the nonlinear optimizer used by OCPBASIC. Therefore it is designed for the use of self defined linear solvers.

## 3.1 Nonlinear optimazation problems in IPbasic

The class of nonlinear optimazation problems, which can be solved by IPBASIC is defined by the following nlp.

**Problem 3.1 (Nonlinear Optimazation Problem)**
*Find the vector* $x \in \mathbb{R}^n$ *to minimize*

$$f(x)$$

*subject to the nonlinear constraints*

$$h(x) = 0$$
$$g(x) \leq 0$$

*with* $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ *and* $g : \mathbb{R}^n \to \mathbb{R}^q$.

The definition of the interior-point-method used by IPBASIC can be found in [1] and [2].

## 3.2 Parameter settings of IPbasic

$\mu_0$:

| | |
|---|---|
| **Notation:** | mu_0 |
| **Description:** | Startvalue of the barrier parameter $\mu$. |
| **Value range:** | $[0, 1]$ |
| **Default value:** | 0.5 |

$\epsilon_{\text{tol}}$:

| | |
|---|---|
| **Notation:** | epsilon_tol |
| **Description:** | Tolerance of the interior-point-method. |
| **Value range:** | $[0, 1]$ |
| **Default value:** | $1e-8$ |

$\Theta^{\mu}$:

| | |
|---|---|
| **Notation:** | Theta_mu |
| **Description:** | Correction parameter of the barrier parameter $\mu$. |
| **Value range:** | $(1, 2)$ |
| **Default value:** | $\frac{2}{3}$ |

$\kappa_{\epsilon}$:

| | |
|---|---|
| **Notation:** | kappa_epsilon |
| **Description:** | Optimazation constant. |
| **Value range:** | $> 0$ |
| **Default value:** | 10.0 |

$\kappa_{\mu}$:

| | |
|---|---|
| **Notation:** | kappa_mu |
| **Description:** | Optimazation constant. |
| **Value range:** | $(0, 1)$ |
| **Default value:** | 0.9 |

$\max_{\text{iter}}$:

| | |
|---|---|
| **Notation:** | max_iter |
| **Description:** | Max number of iterations. |
| **Value range:** | $> 0$ |
| **Default value:** | 1000 |

$\alpha_{\max}$:

| | |
|---|---|
| **Notation:** | alpha_max |
| **Description:** | Max stepsize. |
| **Value range:** | $> 0$ |
| **Default value:** | 1 |

$\alpha_{\text{tol}}$:

| | |
|---|---|
| **Notation:** | tol_alpha |
| **Description:** | Min stepsize. |
| **Value range:** | $> 0$ |

| **Default value:** | $2e-8$ |
|---|---|

$\sigma$**:**

| **Notation:** | sigma |
|---|---|
| **Description:** | Parameter for the armijo-rule. |
| **Value range:** | $(0,1)$ |
| **Default value:** | 0.01 |

$\beta$**:**

| **Notation:** | beta |
|---|---|
| **Description:** | Parameter for the armijo-rule. |
| **Value range:** | $(0,1)$ |
| **Default value:** | 0.9 |

$\tau_{\min}$**:**

| **Notation:** | tau_min |
|---|---|
| **Description:** | Distance parameter for the bounds. |
| **Value range:** | $(0,1)$ |
| **Default value:** | $1e-4$ |

**Printlevel:**

| **Notation:** | PrintLev |
|---|---|
| **Description:** | Printlevel of IPbasic. |
| **Value range:** | **PRINT_NONE** = No printing. |
| | **PRINT_LESS** = Printing only solution. |
| | **PRINT_MORE** = Printing iterations and solution. |
| | **PRINT_MARKED** = Printing iterations and solution with colors. |
| **Default value:** | **PRINT_MORE** |

$\lambda_{\text{tol}}$**:**

| **Notation:** | lambda_tol |
|---|---|
| **Description:** | Tolerance of the lagrange multipliers. |
| **Value range:** | $(0,1)$ |
| **Default value:** | $1e-10$ |

**RegDiagMatrixEqu:**

| | |
|---|---|
| **Notation:** | RegDiagMatrixEqu |
| **Description:** | Regularisation parameter of the diagonalmatrix of the equations. |
| **Value range:** | |
| **Default value:** | 0 |

**RegDiagMatrixInitialGuessLambda:**

| | |
|---|---|
| **Notation:** | RegDiagMatrixInitialGuessLambda |
| **Description:** | Regularisation parameter of the diagonalmatrix lagrange multipliers. |
| **Value range:** | |
| **Default value:** | $1e-9$ |

**CheckMatricesFor_INF_NAN:**

| | |
|---|---|
| **Notation:** | CheckMatricesFor_INF_NAN |
| **Description:** | Tells if matrices are checked due to inf and nan. |
| **Value range:** | bool |
| **Default value:** | true |

**LineSearchCouples:**

| | |
|---|---|
| **Notation:** | MyLineSearchCouples |
| **Description:** | Used LineSearchCouples. |
| **Value range:** | **COUPLES_NONE** = No couples.<br>**COUPLE_STATE_SLACK** =<br>Couple linesearch of slack and state variables.<br>**COUPLE_STATE_SLACK_LAMBDA** =<br>Couple linesearch of slack variables, state variables and lagrange multipliers. |
| **Default value:** | **COUPLE_STATE_SLACK** |

**BreakpointMaxTime:**

| | |
|---|---|
| **Notation:** | BreakpointMaxTime |
| **Description:** | Tells, if there is a breakpoint with a maximal time. |
| **Value range:** | bool |
| **Default value:** | false |

**maximalTime:**

| | |
|---|---|
| **Notation:** | maximalTime |
| **Description:** | If BreakpointMaxTime, maximal time for IPbasic to solve. [in s] |
| **Value range:** | $> 0$ |
| **Default value:** | 1 |

### 3.2.1 Optimizer flags for IPbasic

**LineSearchMethod:**

| | |
|---|---|
| **Notation:** | UsedLineSearchMethod |
| **Description:** | Used linesearch method in IPBASIC. |
| **Value range:** | **LINESEARCH_NONE** = |
| | No linesearch used. |
| | **LINESEARCH_ARMIJO_L1_PENALTY** = |
| | Armijo linesearch with $\ell_1$-penalty-function. |
| **Default value:** | **LINESEARCH_ARMIJO_L1_PENALTY** |

# 4 Examples

Some examples for the use of IPBASIC can be found in the examples/examples_IPbasic/ folder, which is included in OCPbasicMaster.

## 4.1 IPbasic using dense linear solver

For using IPBASIC without OCPBASIC or HSL you will need a linear solver for solving nlps. For small nlps you might want to use a dense linear solver from LAPACK. IPBASIC is desinged for overloading the linear solver. This example will show how a dense linear solver can be defined with IPBASIC and LAPACK_WRAPPER.

### Dense linear solver

IPBASIC is using the SparseMatrixBase class from LAPACK_WRAPPER. To create a dense linear solver you need to overload the LinearSolver_interface class from IPBASIC. A header only definition of a dense linear solver, using linear algebra methods of LAPACK_WRAPPER, can be created by the following code example.

```
using lapack_wrapper;

class DenseSolver : public
IPbasic::LinearSolver_interface<double>
{
lapack_wrapper::LU<double> lu;

public:

DenseSolver()
: LinearSolver_interface<double>()
{
}

virtual ~DenseSolver()
{
}

bool factorize(SparseMatrixBase<double> const & Sparse,
               bool is_stored_symmetric) IPBASIC_OVERRIDE
{
  int const *    pRows;
  int const *    pCols;
  double const * pValues;
  int            numRows, numCols, nnz;
  Sparse.get_info(numRows, numCols, nnz);
  Sparse.get_data(pRows, pCols, pValues);
  lu.allocate(numRows, numCols);
  if (is_stored_symmetric)
  {
    lu.load_sparse_sym(nnz, pValues, pRows, pCols);
  }
  else
  {
    lu.load_sparse(nnz, pValues, pRows, pCols);
  }
  lu.factorize("factorize");
  return true;
```

```
}

bool solve(real_type RHS[]) IPBASIC_OVERRIDE
{
  lu.solve(RHS);
  return true;
}
};
```

### Dense nlp problem

Since the default matrixtype of IPBASIC is using sparse matrices, we should use the lapack_wrapper::SparseCCOOR type for the jacobian and hessian. Still we can use a dense matrix storage layout in LAPACK_WRAPPER. This storage layout is created by the method setup_as_full_column_major(...).

```
class Problem : public IPbasic::NLP<real_type>
{
lapack_wrapper::SparseCCOOR<real_type> Constraint_Jacobian;
lapack_wrapper::SparseCCOOR<real_type> Lagrangian_Hessian;
lapack_wrapper::SparseCCOOR<real_type> MyIPMatrix;
bool                                   Is_equality[2];

public:
Problem()
:NLP<real_type>(&this->Constraint_Jacobian,
&this->Lagrangian_Hessian, &this->MyIPMatrix,
this->Is_equality, false, false)
{
  this->Constraint_Jacobian.setup_as_full_column_major(2, 2);
  this->Lagrangian_Hessian.setup_as_full_column_major(2, 2);
  this->MyIPMatrix.setup_as_full_column_major(4, 4);
  this->is_equality_vec[0] = false;
  this->is_equality_vec[1] = false;
}

int_type state_dimension() const IPBASIC_OVERRIDE
{
  return 2;
```

```
}

virtual void initial_guess(real_type x[]) const
IPBASIC_OVERRIDE
{
  x[0] = 0.6;
  x[1] = 2.0;
}

virtual real_type objective(real_type const x[])
IPBASIC_OVERRIDE
{
  return x[0] * x[0] + x[1] * x[1];
}

virtual void objective_gradient(real_type const x[],
real_type grad[])
IPBASIC_OVERRIDE
{
  grad[0] = 2 * x[0];
  grad[1] = 2 * x[1];
}

virtual int_type constraint_dimension()
const IPBASIC_OVERRIDE
{
  return 2;
}

virtual void constraint(real_type const x[],
real_type c[])
IPBASIC_OVERRIDE
{
  c[0] = -x[1] + 1.5;
  c[1] = -x[1] + exp(x[0]);
}

void constraint_jacobian_update(real_type const x[])
```

```
IPBASIC_OVERRIDE
{
  lapack_wrapper::MatrixWrapper<real_type> M;
  this->Constraint_Jacobian.get_full_view(M);
  M(0, 0) = 0.0;
  M(0, 1) = -1.0;
  M(1, 0) = exp(x[0]);
  M(1, 1) = -1.0;
}

void lagrangian_hessian_update(real_type sigma,
real_type const x[],
real_type const lambda[])
IPBASIC_OVERRIDE
{
  lapack_wrapper::MatrixWrapper<real_type> M;
  this->Lagrangian_Hessian.get_full_view(M);
  M(0, 0) = 2 * sigma + lambda[1] * exp(x[0]);
  M(0, 1) = 0.0;
  M(1, 0) = 0.0;
  M(1, 1) = 2 * sigma;
}
};
```

### Using a dense linear solver

The created dense linear solver can directly be used by IPBASIC. All linear systems are now solved by a dense lu factoization.

```
int main(int, char *[])
{
  Problem     prob;
  DenseSolver MyDenseSolver;

  IPbasic::Solver<double> *      MySolver =
    new IPbasic::Solver<double>(&prob, &MyDenseSolver);
  IPbasic::IPparameter<double> * Param    =
    MySolver->getIPparameter();
  Param->mu_0     = 1e-3;
```

```
  if (MySolver->optimize())
  {
    std::cout << "Job␣done!" << std::endl;
  }
  else
  {
    std::cout << "Job␣not␣done!" << std::endl;
  }

  delete MySolver;
  return 0;
}
```

# References

[1] A. Huber, M. Gerdts, and E. Bertolazzi. Structure exploitation in an interior-point method for fully discretized, state constrained optimal control problems. *Vietnam Journal of Mathematics*, Oct 2018.

[2] A. K.-P. Huber. *Methoden zur Berechnung optimaler Rennlinien im dynamischen Grenzbereich*. PhD thesis, Universität der Bundeswehr München, 2020.