

# DynamicNLPModels

David Cole, Sungho Shin, Francois Pacaud

June 22, 2022

# Contents

<b>Contents</b>	<b>ii</b>
<b>I Introduction</b>	<b>1</b>
1 Introduction	2
2 What is DynamicNLPMODELS?	3
3 Bug reports and support	4
<b>II Quick Start</b>	<b>5</b>
<b>III API Manual</b>	<b>6</b>
4 API Manual	7

## **Part I**

# **Introduction**

## Chapter 1

# Introduction

Welcome to the documentation of [DynamicNLPModels.jl](#)

### **Warning**

This documentation page is under construction.

### **Note**

This documentation is also available in [PDF format](#).

## **Chapter 2**

### **What is DynamicNLPModels?**

## **Chapter 3**

### **Bug reports and support**

Please report issues and feature requests via the [Github issue tracker](#).

## **Part II**

### **Quick Start**

## **Part III**

# **API Manual**



## Chapter 4

# API Manual

[DynamicNLPModels.LQDynamicData](#) - Type.

```
| LQDynamicData{T,V,M,MK} <: AbstractLQDynData{T,V}
```

A struct to represent the features of the optimization problem

$$\underset{u}{\text{minimize}} \frac{1}{2} \sum_{i=0}^{N-1} (s_i^T Q s_i + 2u_i^T S^T x_i + u_i^T R u_i) + \frac{1}{2} s_N^T Q f s_N \text{ subject to } s_{i+1} = A s_i + B u_i \text{ for } i = 0, 1, \dots, N-1, u_i = K s_i + L u_{i-1}$$

---

Attributes include:

- s0: initial state of system
- A : constraint matrix for system states
- B : constraint matrix for system inputs
- Q : objective function matrix for system states from 1:(N-1)
- R : objective function matrix for system inputs from 1:(N-1)
- N : number of time steps
- Qf: objective function matrix for system state at time N
- S : objective function matrix for system states and inputs
- ns: number of state variables
- nu: number of input variables
- E : constraint matrix for state variables
- F : constraint matrix for input variables
- K : feedback gain matrix
- sl: vector of lower bounds on state variables
- su: vector of upper bounds on state variables
- ul: vector of lower bounds on input variables
- uu: vector of upper bounds on input variables
- gl: vector of lower bounds on constraints
- gu: vector of upper bounds on constraints

see also `LQDynamicData(s0, A, B, Q, R, N; ...)`

[source](#)

`DynamicNLPModels.LQDynamicData` – Method.

```
| LQDynamicData(s0, A, B, Q, R, N; ...) -> LQDynamicData{T, V, M, MK}
```

A constructor for building an object of type `LQDynamicData` for the optimization problem

$$\text{minimize } \frac{1}{2} \sum_{i=0}^{N-1} (s_i^T Q s_i + 2u_i^T S^T x_i + u_i^T R u_i) + \frac{1}{2} s_N^T Q f s_N \text{ subject to } s_{i+1} = A s_i + B u_i \forall i = 0, 1, \dots, N-1, u_i = K x_i$$

- 
- `s0`: initial state of system
  - `A`: constraint matrix for system states
  - `B`: constraint matrix for system inputs
  - `Q`: objective function matrix for system states from 1:(N-1)
  - `R`: objective function matrix for system inputs from 1:(N-1)
  - `N`: number of time steps

The following attributes of the `LQDynamicData` type are detected automatically from the length of `s0` and size of `R`

- `ns`: number of state variables
- `nu`: number of input variables

The following keyword arguments are also accepted

- `Qf` = `Q`: objective function matrix for system state at time `N`; dimensions must be `ns` x `ns`
- `S` = nothing: objective function matrix for system state and inputs
- `E` = `zeros(0, ns)`: constraint matrix for state variables
- `F` = `zeros(0, nu)`: constraint matrix for input variables
- `K` = nothing: feedback gain matrix
- `sl` = `fill(-Inf, ns)`: vector of lower bounds on state variables
- `su` = `fill(Inf, ns)`: vector of upper bounds on state variables
- `ul` = `fill(-Inf, nu)`: vector of lower bounds on input variables
- `uu` = `fill(Inf, nu)`: vector of upper bounds on input variables
- `gl` = `fill(-Inf, size(E, 1))`: vector of lower bounds on constraints
- `gu` = `fill(Inf, size(E, 1))`: vector of upper bounds on constraints

[source](#)

`DynamicNLPModels.LQDynamicModel` – Method.

```
| LQDynamicModel(dnlp::LQDynamicData; condense=false) -> LQdynamicModel
| LQDynamicModel(s0, A, B, Q, R, N; condense = false, ...) -> LQDynamicModel
```

A constructor for building a `LQDynamicModel` <: `QuadraticModels.AbstractQuadraticModel` from `LQDynamicData`. Input data is for the problem of the form

$$\text{minimize } \frac{1}{2} \sum_{i=0}^{N-1} (s_i^T Q s_i + 2u_i^T S^T x_i + u_i^T R u_i) + \frac{1}{2} s_N^T Q f s_N \text{ subject to } s_{i+1} = A s_i + B u_i \text{ for } i = 0, 1, \dots, N-1, u_i = K s_i$$

---

If `condense=false`, data is converted to the form

$$\text{minimize } \frac{1}{2} z^T H z \text{ subject to } l_{con} \leq J z \leq u_{con}, l_{var} \leq z \leq u_{var}$$

Resulting `H` and `J` matrices are stored as `QuadraticModels.QPData` within the `LQDynamicModel` struct and variable and constraint limits are stored within `NLPModels.NLPModelMeta`

If `K` is defined, then `u` variables are replaced by `v` variables, and `u` can be queried by `get_u` and `get_s` within `DynamicNLPModels.jl`

---

If `condense=true`, data is converted to the form

$$\text{minimize } \frac{1}{2} u^T H u + h^T u + h_0 \text{ subject to } J z \leq g, l \leq u \leq u$$

Resulting `H`, `J`, `h`, and `h0` matrices are stored within `QuadraticModels.QPData` as `H`, `A`, `c`, and `c0` attributes respectively

If `K` is defined, then `u` variables are replaced by `v` variables. The bounds on `u` are transformed into algebraic constraints, and `u` can be queried by `get_u` and `get_s` within `DynamicNLPModels.jl`

[source](#)

`DynamicNLPModels._build_H` – Method.

```
| _build_H(Q, R, N; Qf = []) -> H
```

Build the (sparse) `H` matrix from square `Q` and `R` matrices such that  $z^T H z = \sum_{i=1}^{N-1} s_i^T Q s_i + \sum_{i=1}^{N-1} u_i^T R u_i + s_N^T Q f s_N$ .

### Examples

```
| julia> Q = [1 2; 2 1]; R = ones(1,1); _build_H(Q, R, 2)
6×6 SparseArrays.SparseMatrixCSC{Float64, Int64} with 9 stored entries:
 1.0  2.0  .  .  .  .
 2.0  1.0  .  .  .  .
 .  .  1.0  2.0  .  .
 .  .  2.0  1.0  .  .
 .  .  .  .  1.0  .
 .  .  .  .  .  .
```

If `Qf` is not given, then `Qf` defaults to `Q`

[source](#)

`DynamicNLPModels._build_sparse_J1` – Method.

```
|_build_sparse_J1(A, B, N) -> J
```

Build the (sparse) J matrix or a linear model from A and B matrices such that  $0 \leq Jz \leq 0$  is equivalent to  $s_{i+1} = A s_i + B s_i$  for  $i = 1, \dots, N-1$

### Examples

```
|julia> A = [1 2 ; 3 4]; B = [5 6; 7 8]; _build_J(A,B,3)
4×12 SparseArrays.SparseMatrixCSC{Float64, Int64} with 20 stored entries:
 1.0  2.0 -1.0  .   .   .   5.0  6.0  .   .   .   .
 3.0  4.0  .  -1.0  .   .   7.0  8.0  .   .   .   .
 .   .   1.0  2.0 -1.0  .   .   .   5.0  6.0  .   .
 .   .   3.0  4.0  .  -1.0  .   .   7.0  8.0  .   .
```

[source](#)

[DynamicNLPModels.get\\_A](#) - Method.

```
|get_A(LQDynamicData)
|get_A(LQDynamicModel)
```

Return the value A from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

[DynamicNLPModels.get\\_B](#) - Method.

```
|get_B(LQDynamicData)
|get_B(LQDynamicModel)
```

Return the value B from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

[DynamicNLPModels.get\\_E](#) - Method.

```
|get_E(LQDynamicData)
|get_E(LQDynamicModel)
```

Return the value E from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

[DynamicNLPModels.get\\_F](#) - Method.

```
|get_F(LQDynamicData)
|get_F(LQDynamicModel)
```

Return the value F from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

[DynamicNLPModels.get\\_K](#) - Method.

```
|get_K(LQDynamicData)
|get_K(LQDynamicModel)
```

Return the value K from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_N` - Method.

```
| get_N(LQDynamicData)  
| get_N(LQDynamicModel)
```

Return the value N from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_Q` - Method.

```
| get_Q(LQDynamicData)  
| get_Q(LQDynamicModel)
```

Return the value Q from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_Qf` - Method.

```
| get_Qf(LQDynamicData)  
| get_Qf(LQDynamicModel)
```

Return the value Qf from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_R` - Method.

```
| get_R(LQDynamicData)  
| get_R(LQDynamicModel)
```

Return the value R from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_S` - Method.

```
| get_S(LQDynamicData)  
| get_S(LQDynamicModel)
```

Return the value S from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_gl` - Method.

```
| get_gl(LQDynamicData)  
| get_gl(LQDynamicModel)
```

Return the value gl from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_gu` - Method.

```
| get_gu(LQDynamicData)  
| get_gu(LQDynamicModel)
```

Return the value gu from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_ns` – Method.

```
| get_ns(LQDynamicData)
| get_ns(LQDynamicModel)
```

Return the value ns from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_nu` – Method.

```
| get_nu(LQDynamicData)
| get_nu(LQDynamicModel)
```

Return the value nu from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_s` – Method.

```
| get_s(solution_ref, lqdm::LQDynamicModel) -> s <: vector
```

Query the solution s from the solver. If `lqdm.condense == false`, the solution is queried directly from `solution_ref.solution`

[source](#)

`DynamicNLPModels.get_s0` – Method.

```
| get_s0(LQDynamicData)
| get_s0(LQDynamicModel)
```

Return the value s0 from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_sl` – Method.

```
| get_sl(LQDynamicData)
| get_sl(LQDynamicModel)
```

Return the value sl from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_su` – Method.

```
| get_su(LQDynamicData)
| get_su(LQDynamicModel)
```

Return the value su from LQDynamicData or LQDynamicModel.dynamic\_data

[source](#)

`DynamicNLPModels.get_u` – Method.

```
| get_u(solution_ref, lqdm::LQDynamicModel) -> u <: vector
```

Query the solution u from the solver. If `K = nothing`, the solution for u is queried from `solution_ref.solution`

[source](#)

`DynamicNLPModels.get_ul` – Method.

```
get_ul(LQDynamicData)
get_ul(LQDynamicModel)
```

Return the value `ul` from `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.get_uu` – Method.

```
get_uu(LQDynamicData)
get_uu(LQDynamicModel)
```

Return the value `uu` from `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_A!` – Method.

```
set_A!(LQDynamicData, row, col, val)
set_A!(LQDynamicModel, row, col, val)
```

Set the value of entry `A[row, col]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_B!` – Method.

```
set_B!(LQDynamicData, row, col, val)
set_B!(LQDynamicModel, row, col, val)
```

Set the value of entry `B[row, col]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_Q!` – Method.

```
set_Q!(LQDynamicData, row, col, val)
set_Q!(LQDynamicModel, row, col, val)
```

Set the value of entry `Q[row, col]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_Qf!` – Method.

```
set_Qf!(LQDynamicData, row, col, val)
set_Qf!(LQDynamicModel, row, col, val)
```

Set the value of entry `Qf[row, col]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_R!` – Method.

```
set_R!(LQDynamicData, row, col, val)
set_R!(LQDynamicModel, row, col, val)
```

Set the value of entry `R[row, col]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_s0!` – Method.

```
set_s0!(LQDynamicData, index, val)
set_s0!(LQDynamicModel, index, val)
```

Set the value of entry `s0[index]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_sl!` – Method.

```
set_sl!(LQDynamicData, index, val)
set_sl!(LQDynamicModel, index, val)
```

Set the value of entry `sl[index]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_su!` – Method.

```
set_su!(LQDynamicData, index, val)
set_su!(LQDynamicModel, index, val)
```

Set the value of entry `su[index]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_ul!` – Method.

```
set_ul!(LQDynamicData, index, val)
set_ul!(LQDynamicModel, index, val)
```

Set the value of entry `ul[index]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)

`DynamicNLPModels.set_uu!` – Method.

```
set_uu!(LQDynamicData, index, val)
set_uu!(LQDynamicModel, index, val)
```

Set the value of entry `uu[index]` to `val` for `LQDynamicData` or `LQDynamicModel.dynamic_data`

[source](#)