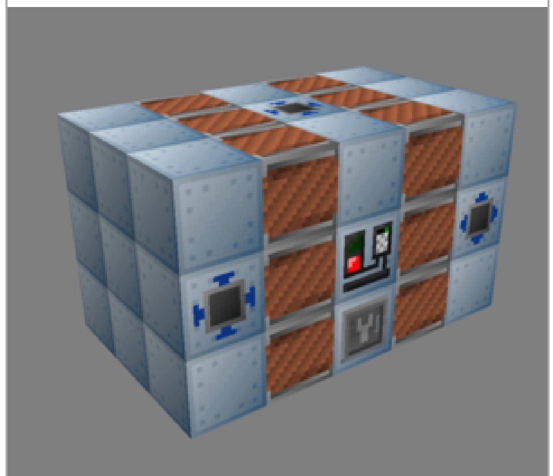
GT New Horizons

Oil Cracking Unit

Introduction

The **Oil Cracking Unit** is a specialized multiblock that uses Steam or Hydrogen to crack oil-based fuels. Compared to a Chemical Reactor, an Oil Cracking Unit gets an additional 25-50% output bonus for the same recipes but runs them at a higher tier with similar EU cost/speed. Steam has a 50% bonus output, Hydrogen 25% bonus and 20% less Hydrogen compared to a Chemical Reactor. The Oil Cracking Unit can be further upgraded with improved coils and has an advanced variant in the Mega Oil Cracker. Beyond power production, this multiblock can make Toluene, a component of Rocket fuel, crack Radox, and break down Naquadah into various noble gasses / radioactive elements.

Oil Cracking Unit



Mod	Gregtech 5
Type	Tile Entity
Tooltip Text	Controller Block for the Oil Cracking Unit
Relevant Quest	Oil Cracker Let's Get Crackin!
Tier	HV
Size	5x3x3
Pollution	None
Properties:	
Blast resistance	6
Hardness	1-5
Energy:	
Energy usage	240+ eu/t
Voltage in	As Energy Hatch
Max amperage	1-2A per Energy Hatch

Construction

An Oil Cracking Unit is a hollow 5x3x3 (WxHxD) structure that requires:

- 1 Controller block; front, centered along one of the 5 block wide sides
- 1 Maintenance Hatch; any casing
- 1+ Energy hatch; any casing
- 1+ Input Hatch for the oil product; any left/right side (1x3x3) ring casings. This must be on the side opposite of the product Output Hatch
- 1+ Output Hatch for the oil product; any left/right side (1x3x3) ring casings. This must be on the side opposite of the product Input Hatch
- 1+ Input Hatch for Steam/Hydrogen input; any middle ring (1x3x3) casings
- 16 identical Coil blocks; forming two hollow vertical rings touching either side of the controller
- 18-21 Clean Stainless Steel Machine Casings; everywhere else

The Oil Cracking Unit can share walls with other Oil Cracking Units or anything that uses the same parts. A single input/output block of each type is recommended as there is very little allowance for extra non-casing blocks. Coils must all be of the same type or an incomplete structure will result.

Usage

The Oil Cracking Unit must have a steady supply of Steam or Hydrogen to work. If there are no Steam Boilers available, a Fluid Heater can make steam from EU and water. IC2 steam is not compatible with this multiblock. Lacking an Input Bus, circuits must be set with a Programmed Circuit item, which is placed in the bottom-right slot of the Controller. Upgraded coils reduce power consumption by 10% per tier, up to a maximum discount of 50% with HSS-G Coil Blocks. In practice, this overclocks and will make recipes run faster for about the same power consumption.

Maximizing Output Calculation and Conclusion

This section is the challenging part of mathematical theory! Here, we will use the language of linear algebra to calculate the best output method, providing a theoretical basis for petrochemical production lines. If you wish to skip the theoretical part, please read through non-maths part and then move to the calculator section below.

We know that each type of desulfurized fuel can be cracked and distilled into chemicals and two other types of fuels. Therefore, after one cracking and distillation, the fuel is not entirely consumed, and we need to calculate the cracking of the remaining fuel by using linear algebra.

The initial approach, which is to maximize the output of the target product from each type of oil shown on NEI, might not be the best combination of cracking. We will represent the proportions of the three types of fuel cracking each other using matrices up to the third order. By using matrix multiplication iteratively and summing infinitely, we can calculate the real maximum output. If heavy fuel is distilled into toluene, the matrix becomes of the second order (the target product should not be toluene). If light fuel is stirred to generate diesel for electricity, the matrix degenerates to a scalar.

This article is only a theoretical calculation, valuable for reference, with lots of detailed elements not considered. Please still try to design your own petrochemical line that suits your own save.

The following mathematical theory may be too hard to comprehend. With the general principle understood, it's suggested to directly use the calculator below if you're not into complicated maths.

Variable Definitions and Interpretations

First, let's define each variable:

κ: The ratio of the final product from cracking and distillation per barrel of crude oil under a specific combination of cracking methods.

δ: The crude oil distillation ratio.

A: The cracking matrix for the three types of fuel under a specific combination, where each element represents the proportion of one fuel type converting to another. Each ROW correspond to a part of a single recipe in NEI.

Note that in the new version, when using steam cracking distillation, each element of A must be multiplied by a factor of 1.2.

α: The yield rate of the target product from each type of fuel under a specific cracking combination.

Here, κ is a scalar, while δ and α are column vectors up to three dimensions. Note that we generally use the row vector of δ^T.

If the definitions seem to be not clear enough, just download the calculator.

Calculation Formula

Now, let's give the specific calculation formula. After inputting a certain amount (a large amount) of a type of crude oil, the first cracking and distillation are performed:

The conversion ratio of the target product in this instance is $\kappa_0 = \delta^T \alpha$, and the remaining fuel ratio is $\delta^T A$

In the second instance, the remaining fuel undergoes further cracking and distillation.

The conversion ratio of the target product up to this point is $\kappa_1 = \kappa_0 + \delta^T A\alpha = \delta^T (I + A)\alpha$, and now the remaining fuel ratio is $\delta^T A^2$

This process can be repeated many times, theoretically infinitely, leading to $\kappa_n = \delta^T \sum_{i=0}^{n} A^i \alpha$

Taking n to infinity, we get $\kappa = \delta^T \sum_{n=0}^{\infty} A^n \alpha$

The infinite sum is similar to a geometric series, and using Taylor's theorem for multivariate functions, we can obtain $\sum_{n=0}^{\infty} A^n = (I - A)^{-1}$

(Note: The negative exponent represents the matrix inversion operation.)

Thus, the formula simplifies to $\kappa = \delta^T (I - A)^{-1} \alpha$

We need to maximize κ under all conditions that meet the constraints (such as not using hydrogen cracking or heavy cracking, etc.). Please note, the type of oil might affect the outcome, but we have not encountered this situation yet.

You can also modify the matrix to get results under different constraints.

For example, when stirring to make diesel but not directly distilling heavy fuel, the fuel cracking matrix A changes. This situation might occur when using heavy oil for electricity generation and cracking heavy fuel to obtain petrochemical by-products in the early to mid-stages. In such cases, the light fuel output in the matrix should be set to zero, and the heavy fuel should be reduced by 0.2 times the light fuel output. Similarly, in δ (crude oil distillation), the light fuel output and 0.2 times the light fuel output in heavy fuel should be removed.

For example, in the matrix of the example, if this operation is performed, then:

$$\delta = \begin{pmatrix} 0.4 \\ 1-1 \\ 0.3 - 0.2 \times 1 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0 \\ 0.1 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 0.1 - 0.1 & 0.125 - 0.1 \times 0.2 \\ 0 & 0 - 0 & 0.1 - 0 \times 0.2 \\ 0.05 & 0 - 0.05 & 0 - 0.05 \times 0.2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0.105 \\ 0 & 0 & 0.1 \\ 0.025 & 0 & -0.01 \end{pmatrix}$$

(in old version without buffs on OCU)

As light fuel does not produce any chemical products, we can eliminate the second row and second column, making it a second-order matrix and completely ignoring light fuel in subsequent calculations (as it's no longer present). But this doesn't really matter.

Calculator

Well, there's a calculator which was mainly contributed by core.exe: GTNHPetroOptim (https://github.com/core-exe/GTNHPetroOptim)

It needs Python to run. Feel free to manipulate the parameters in the code and derived the result you want by yourself.